# LOAN DOCUMENT

PHOTOGRAPH THIS SHEET

DTIC ACCESSION NUMBER

LEVEL

INVENTORY

DOCUMENT IDENTIFICATION

DISTRIBUTION STATEMENT

ACCESSION FOR
NTIS GRA&I
DTIC TRAC
UNANNOUNCED
JUSTIFICATION

BY

DISTRIBUTION/

AVAILABILITY CODES

| DISTRIBUTION | AVAILABILITY AND/OR SPECIAL | |
|---|---|---|
| A-1 | | |

DISTRIBUTION STAMP

DATE ACCESSIONED

DATE RETURNED

HANDLE WITH CARE



19981223 007

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NUMBER

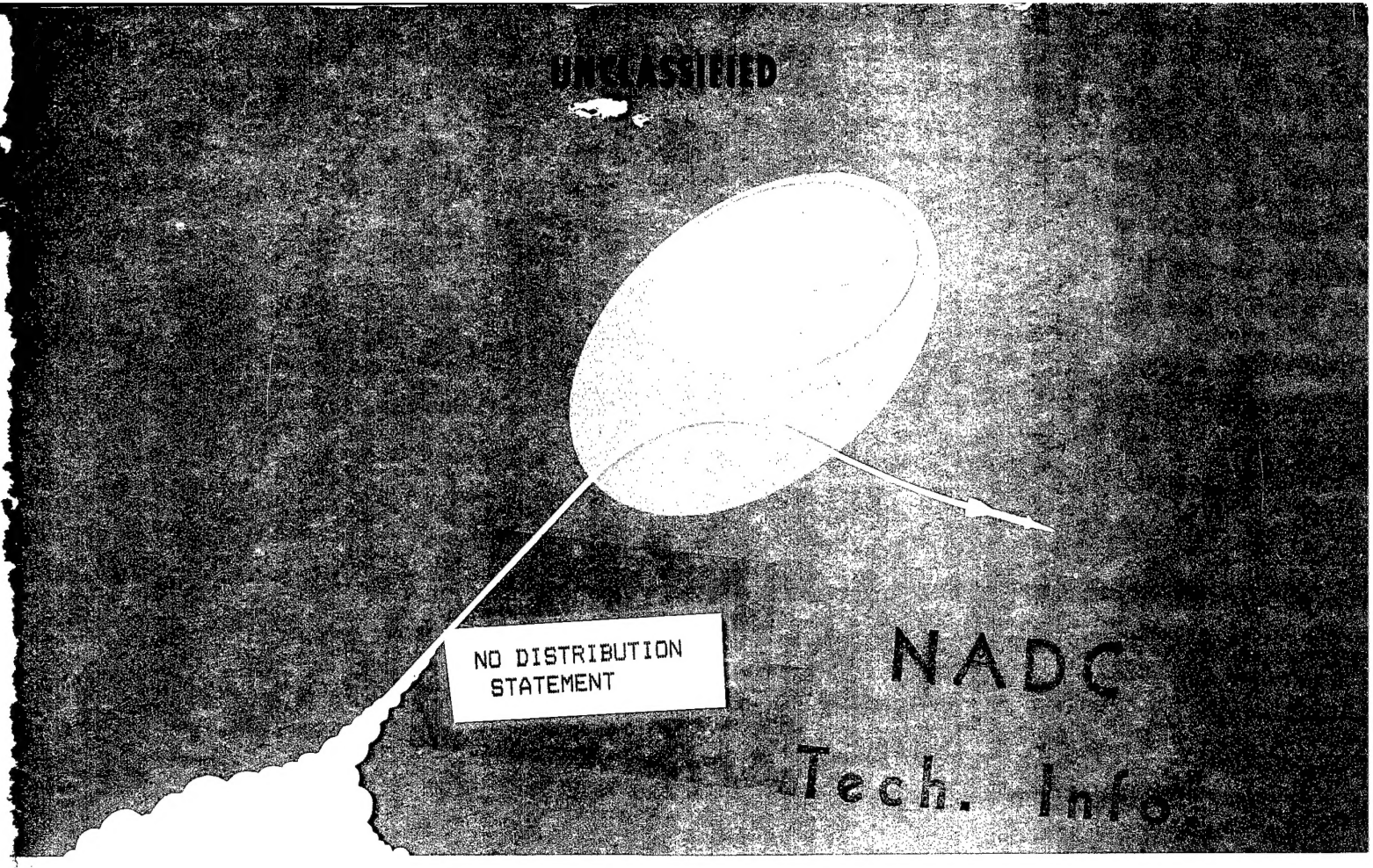
PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC

DTIC FORM 70A JUN 90

DOCUMENT PROCESSING SHEET

PREVIOUS EDITIONS MAY BE USED UNTIL STOCK IS EXHAUSTED.

LOAN DOCUMENT




APPENDIX 6

SYSTEM CONTROLLER EXECUTIVE DESIGN SPECIFICATION

FINAL SOFTWARE REPORT

DATA ITEM NO. A005

# INTEGRATED ELECTRONIC WARFARE SYSTEM
# ADVANCED DEVELOPMENT MODEL (ADM)



PREPARED FOR:

NAVAL AIR DEVELOPMENT CENTER
WARMINSTER, PENNSYLVANIA

CONTRACT N62269-75-C-0070

RAYTHEON
ELECTROMAGNETIC
SYSTEMS DIVISION

1 OCTOBER 1977

| RAYTHEON | RAYTHEON COMPANY<br>LEXINGTON, MASS. 02173 | CODE IDENT NO.<br>49956 | SPEC NO.<br>53959-GT-0756 |
|---|---|---|---|
| | | | SHEET 1 OF 78 / REV A |

TYPE OF SPEC

# COMPUTER SUBPROGRAM DESIGN DOCUMENT

TITLE OF SPEC

# SYSTEM CONTROLLER EXECUTIVE, IEWS, ADM

| FUNCTION | APPROVED | DATE | FUNCTION | APPROVED | DATE |
|---|---|---|---|---|---|
| WRITER | R. R. Scott | 12 Oct 76 | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

REVISIONS

| CHK | DESCRIPTION | REV | CHK | DESCRIPTION | REV |
|---|---|---|---|---|---|
| TLC | Complete Rewrite - 25 May 1977 | A | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REVISION | A | | | | | | | | | | | | | | | | | | | | | | | |
| SHEET NO. | | | | | | | | | | | | | | | | | | | | | | | | |
| REV STATUS OF SHEETS | REVISION | | | | | | | | | | | | | | | | | | | | | | | |
| | SHEET NO. | | | | | | | | | | | | | | | | | | | | | | | |

10-1348 (11/68) VELLUM PRINTED IN U.S.A.
10-1348 (11/68) FILM

# TABLE OF CONTENTS

# 1.0 SCOPE

## 1.1 IDENTIFICATION

This document specifies the detailed design requirements of the System Controller (SC) Executive Function (EXEC) subprogram for the Advanced Development Model (ADM) of the Integrated Electronic Warfare System (IEWS).

## 1.2 SUBPROGRAM TASKS

A copy of the EXEC shall reside in each of the following SC processors:

1. Resource Management (RMP)
2. Classification (CP)
3. Analysis (AP)

The mnemonics assigned to the three EXEC's shall be, respectively:

1. EXRM
2. EXCP
3. EXAP

Each EXEC shall be the primary control module for the operational software in each of the three processors. Each EXEC and the ECM processing modules assigned to that EXEC shall constitute a real-time operating system with priority multi-tasking capability. The three EXEC's shall communicate with each other to initiate and coordinate ECM processing in each of the three processors. The three EXEC's shall be similar in design. Differences among the three shall be as specified in this document.

In order to perform the tasks required for the servicing and control of ECM processing modules, the Executive function shall be divided into five subfunctions:

1) Initialization sequence (EXINT)

2) Interrupt handlers

   (a) RMP: Bus hung (EXHNG)
   Real-time clock (EXTIM)

   (b) CP: Bus hung (EXHNG)
   Real time clock (EXTIM)
   Sorter power fail (EXSPF)
   Sorter high-priority message (EXIH)

   (c) AP: Bus hung (EXHNG)
   Real time update from RMP (EXTIM)
   Aux Bus Interface buffer full (EXBF)

3) Task manager (ECM Processing Manager)

   (a) Scheduler (EXSCD)

   (b) Dispatcher (EXDIS)

4) Executive message handler

   (a) ECM-to-Exec interface (EXMES or EXMSG)

   (b) Executive message processing routine (XMES)

   (c) Intraprocessor communication message blocks (MTB's)
   Manager (GTMN and FRMN)

5) Interprocessor communication interface

   (a) Message receiver (EXIPR)

   (b) Message sender (EXIPS)

   (c) Interprocessor communication memory blocks (MCB's)
   Manager (GTMNC and FRMNC)

   (d) Special interprocessor communication (EXSSS, EXSTE)

## 2.0 REFERENCE DOCUMENTS

### 2.1 PERFORMANCE SPECIFICATIONS

The CPPS for the System Controller Unit, Integrated Electronic Warfare System (IEWS), Advanced Development Model (ADM), Raytheon Document No. 061290529, paragraph 3.3.1 shall apply to this subprogram.

### 2.2 DESIGN SPECIFICATIONS

The CPDS for the System Controller Program, IEWS, ADM Specification No. 53959-GT-0759, Section 3, paragraph 3.2.2, Table IIA, and paragraphs 3.2.3., 3.2.3.1, and 3.2.3.2 shall apply to this subprogram.

### 2.3 DATA BASE DESIGN DOCUMENT

The Common Data Base Design Document, System Controller Unit, IEWS, ADM, Document No. 53959-GT-0751, shall apply to this subprogram.

### 2.4 MISCELLANEOUS DOCUMENTS

The following documents shall apply to the subprogram:

| Document No. | Document Title |
|---|---|
| WS-8506, Revision 1, 1 November 1971 | Requirements for Digital Computer Program Documentation |

# 3.0 REQUIREMENTS

## 3.1 SUBPROGRAM DETAILED DESCRIPTION

### 3.1.1 Introduction

IEWS System Controller operational software shall consist of ECM processing software and control software. ECM processing software is the implementation of the algorithms required to analyze and react to the ECM environment. Control software (i.e., the Executive) is the software that:

1) shall initialize all operational software and hardware
2) shall respond to interrupts generated by IEWS hardware units (e.g., Real Time Clock) and subsystems (e.g., Signal Sorter)
3) shall provide control of ECM processing modules (task management)
4) shall allow the three SC processors (RMP, CP, and AP) to communicate with each other
5) shall allow the SC to communicate with the Signal Sorter (SS) and Special Test Equipment (STE)
6) shall allow the ECM processing software to communicate with the control software.

The control software shall maintain control of the ECM processing through the use of interrupts from external devices (e.g., Signal Sorter, Real Time Clock, etc.). Control software (i.e., the Executive) shall also be known as "foreground processing". The ECM processing software shall also be known as "background processing", because it is executed at the request of the foreground and because it can be interrupted by the foreground.

The Executive function (i.e., the control software) shall consist of five subfunctions:

1) Initialization Sequence (EXINT)
2) Interrupt Handlers
3) ECM Processing Management (EXSCD and EXDIS)
4) Executive Message Handler (EXMES)
5) Interprocessor Communications Management (EXIPR and EXIPS)

The overall structure of each of the three Executives is shown in Figures 1, 2, and 3 (RMP, CP, and AP respectively). Rectangular blocks represent software modules and circular blocks represent data structures.

### 3.1.2 Initialization Sequence (EXINT)

EXINT shall be the starting address for the stored operational program in the RMP, CP, and AP. Control shall be sent to EXINT in the RMP (the master processor in the SC) by the SC loader, after the loader has completed the loading of all IEWS operational software. Control shall be sent to EXINT in the CP and AP by the RMP's initialization sequence. The three initialization sequences are significantly different. However, the operations performed by each sequence can be categorized as follows:

1) Set stack register to initial value
2) Initialize (i.e., halt) all slave processors
3) Initialize interprocessor communication data (MCB's free queue, lockout flags, polling flags, and chain pointers)
4) Initialize intraprocessor message blocks (MTB's) free queue
5) Initialize common data base items
6) Initialize hardware units
7) Initialize interrupt trap memory locations
8) Initialize task management queues
9) Newstart (put into run mode) all slave processors

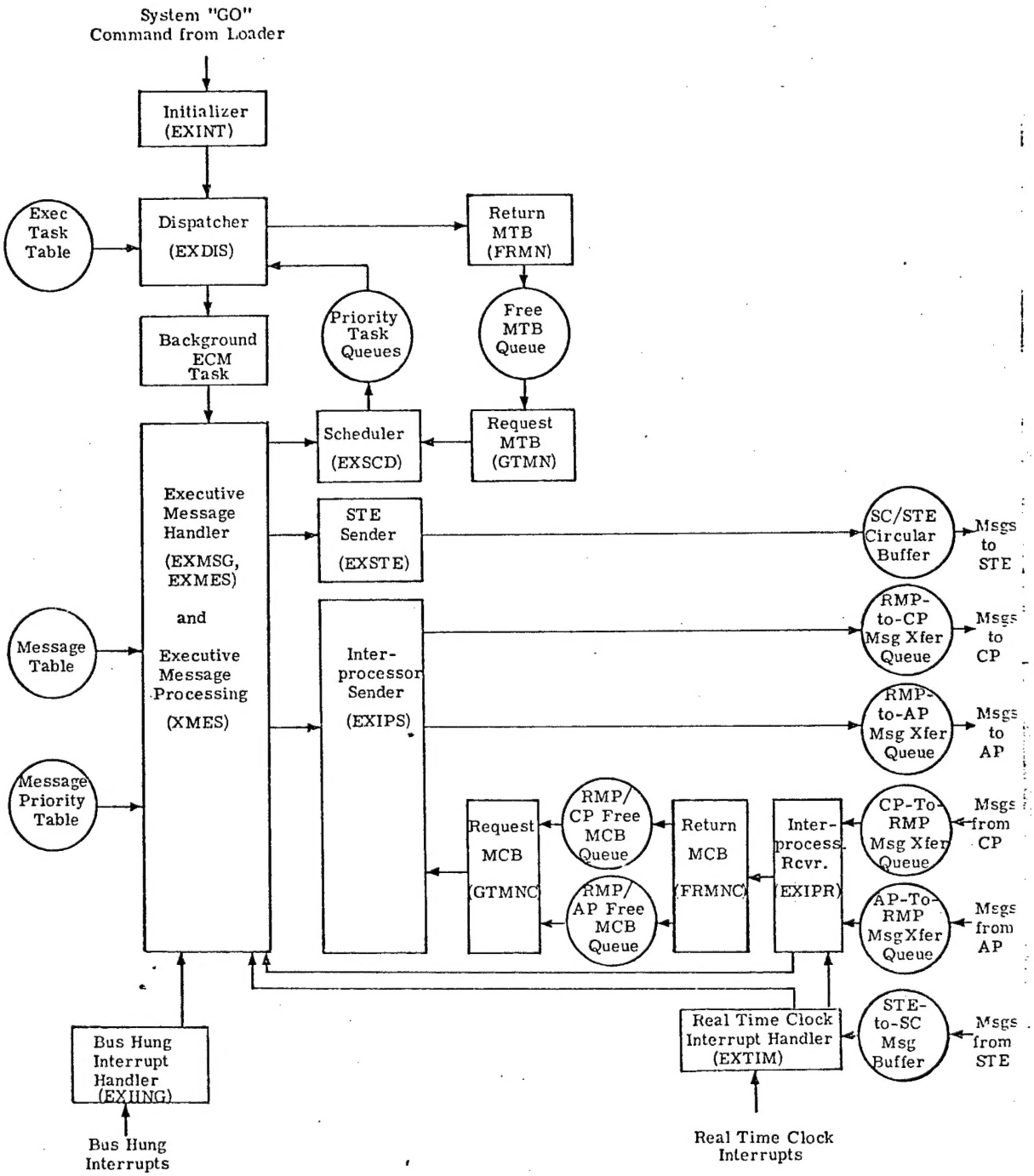



Figure 1. RMP EXEC Block Diagram

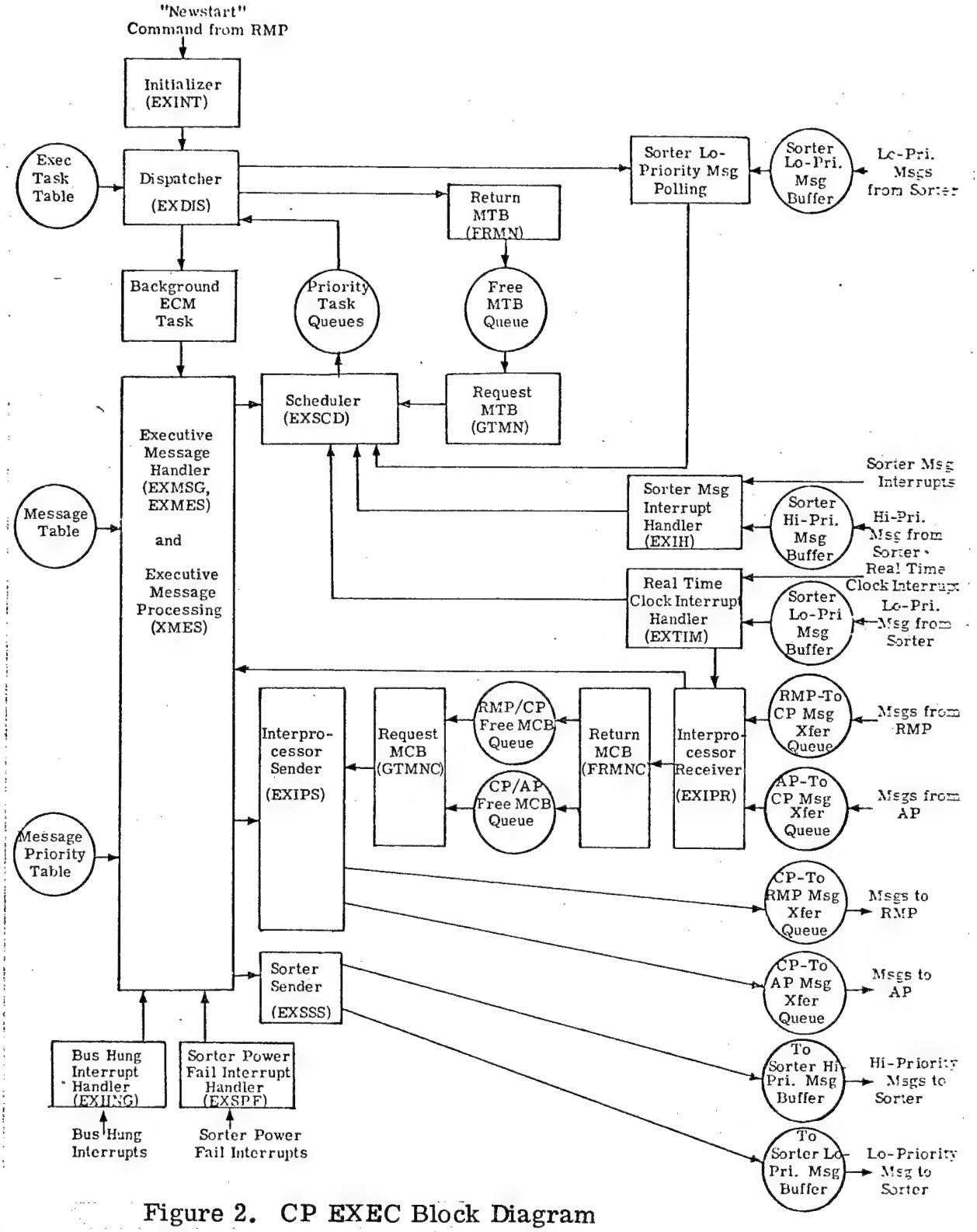



Figure 2. CP EXEC Block Diagram

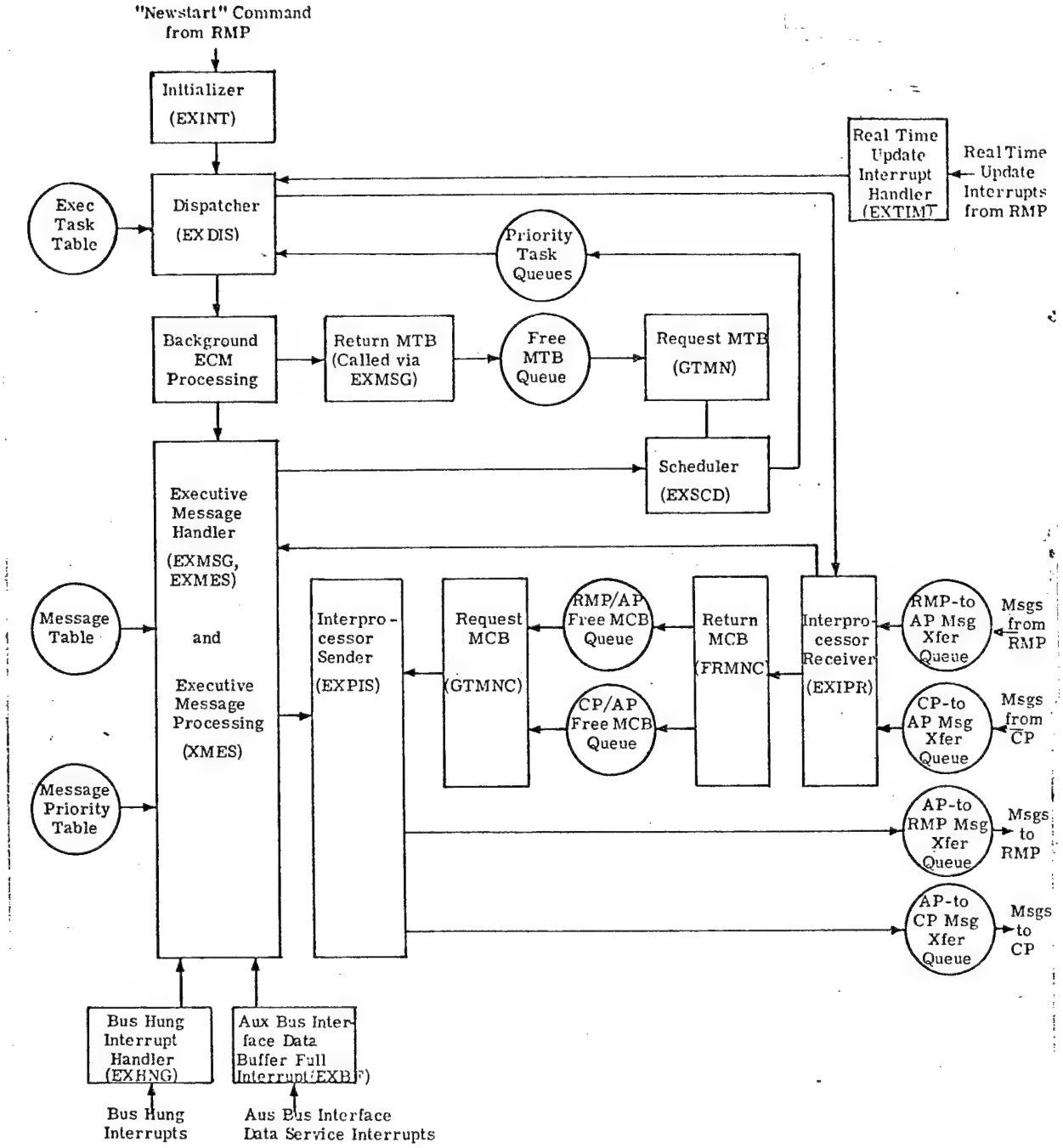



Figure 3. AP EXEC Block Diagram

10) Enable all interrupts

11) Exit to Executive dispatcher (EXDIS)

The three initialization sequences shall be specified by the following tables:

1) RMP Initialization Table 1

2) CP Initialization Table 2

3) AP Initialization Table 3

3.1.3 Interrupt Handlers

Interrupt handlers shall be the set of routines immediately called in response to an interrupt in the RMP, CP, or AP. The addresses of the interrupt handlers for a given processor shall be loaded by EXINT into the interrupt trap addresses assigned in that processor. The contents of the trap addresses are used as indirect addresses by the RP-16 to call the appropriate interrupt handler when the interrupt occurs. Tables 4, 5, and 6 define the priority interrupt structure in the RMP, CP, and AP respectively.

Each interrupt handler is unique, but the general structure shall be:

1) Save A, B, E, and X-register contents on stack.

2) Take action appropriate to the interrupt (usually the scheduling of a driver, i.e., task)

3) Restore A, B, E, and X-register contents

4) Re-enable the interrupt

5) Return to program executing at the time of interrupt.

It should be noted that the contents of the overflow flip-flop is not saved by the interrupt handlers. If this flip-flop is to be tested (Jump on overflow instruction), the following sequence of instructions shall be required:

# TABLE 1

## RMP INITIALIZATION SEQUENCE

| Item No. | Description |
|---|---|
| 1. | Set S-register to high address of memory reserved for stack |
| 2. | Initialize Technique Generator Processor (TG) |
| 3. | Initialize CP |
| 4. | Initialize AP |
| 5. | Link together interprocessor message blocks (MCB's) used in RMP/CP communication |
| 6. | Clear RMP/CP interprocessor communication lockout flags, polling flags, chain pointers, next link pointers |
| 7. | Link together interprocessor message blocks (MCB's) used in RMP/AP communication |
| 8. | Clear RMP/AP interprocessor communication lockout flags, polling flags, chain pointers |
| 9. | Link together intraprocessor message blocks (MTB's) |
| 10. | Clear Azimuth Link Table (AZ) |
| 11. | Clear IEWS System Time<br>(Systim 2, Systim 1- RMP/CP copy)<br>(Systim 4, Systim 3 - RMP/AP copy) |
| 12. | Initialize encoding threshold current value to maximum value (value of SYTHU) |
| 13. | Initialize Jam Status File (JS)<br>Files 0, 2, 13, 15: Inactive<br>All Others: Active |
| 14. | Clear Resource File (RF) |
| 15. | Initialize Display/Control Status File (CD) All words cleared except (CD + 2) x '8000' |
| 16. | Clear Polar Image File (PI) |
| 17. | Clear Alphanumeric Image (AN) |
| 18. | Initialize Alphanumeric Memory pointer (ACPTR) - to-start address of Alphanumeric memory (AC) |

TABLE 1

(Continued)

| Item No. | Description |
|---|---|
| 19. | Unit clear Emitter Tracker (ET) |
| 20. | Unit clear Parameter Encoder (PE) |
| 21. | Call System Management 2 Driver to interrogate INS and transfer aircraft parameters of heading, altitude, pitch and roll to Common Data Base and store encoding threshold current value and azimuth correction factor in SC/PE Interface |
| 22. | Clear program load lamp on polar display |
| 23. | Initialize Serial I/O Channel 2 (Alphanumeric display) |
| 24. | Store addresses of 16 interrupt handlers in memory trap locations |
| 25. | Unmask (via PIN/RTC mask register) interrupt levels<br>1 Bus Hung<br>2 Real Time Clock |
| 26. | Reset Real Time Clock counter and set interval to 2 msec |
| 27. | Enable hung bus detector |
| 28. | Clear task management queues |
| 29. | Store address of watchdog timer expiration routine (EXWDG) in watchdog trap address and enable watchdog timer |
| 30. | Newstart TG |
| 31. | Newstart AP |
| 32. | Newstart CP |
| 33. | Enable high level interrupt |
| 34. | Enable low level interrupt |
| 35. | Unconditional exit to executive dispatcher (EXDIS) |

TABLE 2

CP INITIALIZATION SEQUENCE

| Item No. | Description |
|---|---|
| 1. | Set S-Register to high address of memory reserved for stack |
| 2. | Initialize Signal Sorter |
| 3. | Link together interprocessor message blocks (MCB's used in CP/AP communication |
| 4. | Clear CP/AP interprocessor communication lockout flags; polling flags, chain pointers, next link pointers |
| 5. | Link together intraprocessor message blocks (MTB's) |
| 6. | Set amplitude threshold constant (ATC) to 5 |
| 7. | Clear From - SS high priority message interrupt |
| 8. | Store addresses of 16 interrupt handlers in memory trap locations |
| 9. | Initialize Emitter Track File (ETF) by calling SOIE (Initialize 1 ETF entry) 128 times. |
| 10. | Unmask (via PIN/RTC mask register) interrupt levels<br>1 Bus hung<br>2 Real Time Clock<br>3 Signal Sorter Power Fail<br>4 Signal Sorter Hugh Priority message |
| 11. | Reset Real Time Clock counter and set interval to 2 msec |
| 12. | Enable hung bus detector |
| 13. | Clear task management queues |
| 14. | Newstart Signal Sorter |
| 15. | Send "Sorter Start" message to Signal Sorter |
| 16. | Enable high level interrupt |
| 17. | Enable low level interrupt |
| 18. | Unconditional exit to executive dispatcher (EXDIS) |

TABLE 3

AP INITIALIZATION SEQUENCE

| Item No. | Description |
|---|---|
| 1. | Set S-Register to high address of memory reserved for stack |
| 2. | Clear AP/CP interprocessor communication next link pointers |
| 3. | Link together intraprocessor message blocks (MTB's) |
| 4. | Call ABI Management 2 initializer (AB2IN) |
| 5. | Set value of interrupt counter (CTR) to 5 |
| 6. | Store addresses of 2 interrupt handlers in memory trap locations |
| 7. | Clear task management queues |
| 8. | Enable high level interrupt |
| 9. | Enable low level interrupt |
| 10. | Unconditional exit to executive dispatcher (EXDIS) |

TABLE 4

RMP INTERRUPT STRUCTURE

| Hex Trap Address | Interrupt Handler Mnemonic | HI of LO Priority | Interrupt Name | Comments |
|---|---|---|---|---|
| 9000 | NOTUSD | LO, Ø | Wired to output interrupt Ø | Used for test purposes only. High level interrupt is unused. |
| 9008 | EXHNG | LO, 1 | Bus Hung Detector | PIN/RTC module vectors low level interrupts. |
| 9010 | EXTIM | LO, 2 | Real Time Clock | |
| 9018 | NOTUSD | LO, 3 | CP-sourced interrupt | Not used in IEWS, ADM |
| 9020 | NOTUSD | LO, 4 | AP-sourced interrupt | " |
| 9028 | NOTUSD | LO, 5 | Serial I/O Ch 1 | " |
| 9030 | NOTUSD | LO, 6 | Serial I/O Ch 2 | " |
| 9040 | NOTUSD | LO, 8 | Serial I/O Ch 4 | " |
| 9048 | NOT USD | LO, 9 | Daisy Chain | " |
| 9050 | NOTUSD | LO, 10 | Daisy Chain | Not used in IEWS, ADM |
| 9058 | NOTUSD | LO, 11 | Daisy Chain | " |
| 9060 | NOTUSD | LO, 12 | Daisy Chain | " |
| 9068 | NOTUSD | LO, 13 | Daisy Chain | " |
| 9070 | NOTUSD | LO, 14 | Daisy Chain | " |
| 9078 | NOTUSD | LO, 15 | Daisy Chain | " |
| 9038 | NOTUSD | LO, 7 | Serial I/O Ch 3 | " |

TABLE 5

CP INTERRUPT STRUCTURE

| Hex Trap Address | Interrupt Handler Mnemonic | HI or LO Priority | Interrupt Name | Comments |
|---|---|---|---|---|
| 0 | NOTUSD | HI | RMP-sourced interrupt | Not used in IEWS, ADM |
| 8 | EXHNG | LO, 1 | Bus Hung detector | PIN/RTC module vectors low level interrupts |
| 10 | EXTIM | LO, 2 | Real Time Clock | |
| 18 | EXSPF | LO, 3 | Signal sorter power fail | |
| 20 | EXIH | LO, 4 | Signal sorter message | |
| 28 | NOTUSD | LO, 5 | Serial I/O Ch 5 | Not used in IEWS, ADM |
| 30 | NOTUSD | LO, 6 | Serial I/O Ch 6 | Not used in IEWS, ADM |
| 38 | NOTUSD | LO, 7 | Wired to output Interrupt Ø | Not used in IEWS, ADM (Used for test purposes only) |
| 40 | NOTUSD | LO, 8 | Unused | Not used in IEWS, ADM |
| 48 | NOTUSD | LO, 9 | " | " |
| 50 | NOTUSD | LO, 10 | " | " |
| 58 | NOTUSD | LO, 11 | " | " |
| 60 | NOTUSD | LO, 12 | " | " |
| 68 | NOTUSD | LO, 13 | " | " |
| 70 | NOTUSD | LO, 14 | " | " |
| 78 | NOTUSD | LO, 15 | " | " |

TABLE 6

AP INTERRUPT STRUCTURE

| Trap Address | Interrupt Handler Mnemonic | HI or LO, Priority | Interrupt Name | Comments |
|---|---|---|---|---|
| 0 | EXTIM | HI | RMP - sourced clock interrupt | Generated by RMP every 10 msec. |
| 1 | EXIH | LO | All of the following low level interrupts:<br>1. Bus hung detector (EXHNG)<br>2. ABI Data Buffer Full (EXBF) | No PIN/RTC module is resident in AP. Interrupt vectoring is performed by EXIH. |

1. Disable interrupts
2. Reset overflow flip-flop
3. Perform operation which may set overflow flip-flop (e.g., divide)
4. Test overflow flip-flop
5. Enable interrupts

3.1.4 Task Manager (EXSCD and EXDIS)

The task manager, which shall also be known as the ECM processing manager, shall consist of two routines:

1. EXSCD: The Executive Scheduler Subroutine
2. EXDIS: The Executive dispatcher

3.1.4.1 Executive Scheduler Subroutine - EXSCD shall be an internal routine (called only by executive routines, i.e., control software). Its function shall be to place a message transfer block (MTB) on one of the priority task queues. The MTB's in this context can be thought of as task control blocks. Word Ø of the MTB shall always be an executive message number (see CDBDD) when the MTB is being used as a task control block. The executive message number shall be used to call the appropriate driver (task) when the driver is to be dispatched. There shall be two priority task queues. All requests for ECM processing shall be placed on the low priority queue with the following exceptions:

1. In the CP, New Emitter Alert messages received by EXIH from the Signal Sorter via the high-priority message buffer and destined for Sorter Message Processing (SODR), and
2. In the AP, requests to execute the Analysis Return Driver (ABRDR) generated by the receipt of Aux Bus Data Buffer Full interrupt by EXBF.

3.1.4.2 Executive Dispatcher - The function of EXDIS shall be to search the priority task queues for the next request for ECM processing and to transfer control to the driver assigned to servicing that request. The algorithm for determining the next request to be serviced shall be as follows:

1. All requests on the high priority queue shall be dispatched before a request on the low priority queue can be serviced.
2. Within priorities, servicing shall be on a FIFO basis.

Control shall be transferred to EXDIS:

1. At the completion of EXINT
2. After the completion of an ECM processing task (Executed as a subroutine)
3. When the task queues are empty

The CP version of EXDIS shall poll the low priority message buffer from the Signal Sorter as its idle loop. That is, when the CP task queues are empty, the message buffer shall be polled before the queues are re-searched. If a message from the Sorter is present, an MTB shall be obtained (via GTMN), the message copied from the low priority buffer to the MTB, and the Sorter Message Processing Driver (SODR) scheduled.

3.1.5 Executive Message Handler (EXMES, XMES, GTMN, FRMN)

The Executive message handler shall be the Executive module responsible for processing all communication sent from ECM processing to the Executive. The Executive message handler shall then decide if the message is interprocessor in nature, intraprocessor, or special, and take the appropriate action.

3.1.5.1 Executive Message Handler (EXMES or EXMSG) - EXMES shall receive from the ECM processing routine the address of an executive message in the X-register. EXMES shall simply:

1. Disable interrupts
2. Call XMES to do the actual message processing
3. Enable interrupts
4. Return to calling routine

3.1.5.2 Executive Message Processing (XMES) - XMES shall be the message switching routine of the Executive message handler. XMES shall only be called from the control software (EXEC). XMES shall be driven by two tables:

1. MTAB, the message routing table, and
2. PTAB, the message priority table.

There shall be an entry (one 16-bit word) in each of these tables for each Executive message type. XMES shall get the message number from the incoming message and get the corresponding MTAB entry. If the MTAB entry is all ones, the incoming message is destined for another processor (RMP, CP, or AP). EXIPS shall be called to perform the processing to that end. If the MTAB entry is zero, the incoming message is destined for a driver in the same processor. To do the message switching:

1. An MTB shall be obtained from the queue of available MTB's (via a call to GTMN)
2. The incoming message shall be copied to the MTB.
3. The MTB is placed on one of the priority task queues, using the priority obtained from PTAB (call to EXSCD)

If the MTAB entry is not all ones and not zero, it is used as an address to which control shall be sent to do special processing. Special processing shall include:

1. Request (Executive message type 24) and release (Executive message type 25) of MTB's for ECM processing (i.e., Candidate List management in CP, task control block release in AP).

2. Event flag setting/resetting.
   (a) Alphanumeric update flag setting in the RMP (Executive message type 18)
   (b) Display flag setting in the RMP
3. Routing of Executive message type 5 from the RMP to the STE and routing of Executive message type 19 from the CP to the Signal Sorter.

3.1.5.3 MTB Management (GTMN and FRMN) - The memory blocks used for intraprocessor task scheduling and messages (MTB's) shall be controlled by the Get Main (GTMN) and Free Main (FRMN) subroutines. The MTB's shall be linked by EXINT with forward pointers to the next MTB to form a queue of available MTB's. GTMN uses the start of queue pointer (XSTR) to remove MTB's from the free queue. FRMN uses the end of queue pointer (XEND) to return MTB's to the free queue.

3.1.6 Executive Interprocessor Communication (EXIPR, EXIPS)

EXIPR and EXIPS shall have the primary function of conducting all interprocessor communication between the resource, analysis, and classification processors.

This process shall be accomplished in the following manner. A copy of EXIPR and EXIPS shall be resident in each of the above specified processors. Communication buffers shall be stablished in each of the common memories between processors. The buffers shall be segmented into message control blocks (MCB's). The MCB's shall be linked via a linked list.

The MCB's are allocated upon demand with a get MCB request (call to GTMNC) and returned to free status via a free MCB request (call to FRMNC).

3.1.6.1 Interprocessor Communication Receiver (EXIPR) - EXIPR shall determine whether there are any messages for transfer by checking the poll flags for each communication buffers. If there are no messages, i.e., the poll flags are clear - the routine returns. If there are messages to transfer (indicated by a non-zero poll):

1. The pointer to the first MCB shall be loaded.
2. The poll shall be decremented.
3. The message shall be processed by calling XMES.
4. The next link shall be examined. If the next link is not zero, the above steps shall be repeated until a zero link is encountered.
5. The MCB shall be returned to free storage (via a call to FRMNC).

3.1.6.2 Interprocessor Communication Sender (EXIPS) - EXIPS shall have the responsibility of directing messages from one of the three SC processors (RMP, CP, or AP) to either of the other two. EXIPS shall be driven by the Message/Processor Table (SMSPC). SMSPC shall have one 16-bit word entry for each Executive message type. The contents of each entry shall be used to decide to which processor the incoming message is to be sent. For example, in the RMP, a Ø-entry directs a message to the CP and 1-entry directs it to the AP.

After the destination processor has been selected, EXIPS shall request the allocation of the appropriate MCB and the message shall be copied into it. The poll is checked:

If zero: The pointer is loaded with the MCB's address. The poll is incremented by one.

If non-zero: The address of the current MCB is placed in the contents of the address indicated by the next link location. The poll is incremented.

Control is returned to the caller.

An example of one direction of the interprocessor communication algorithm is shown in Figure 3a.

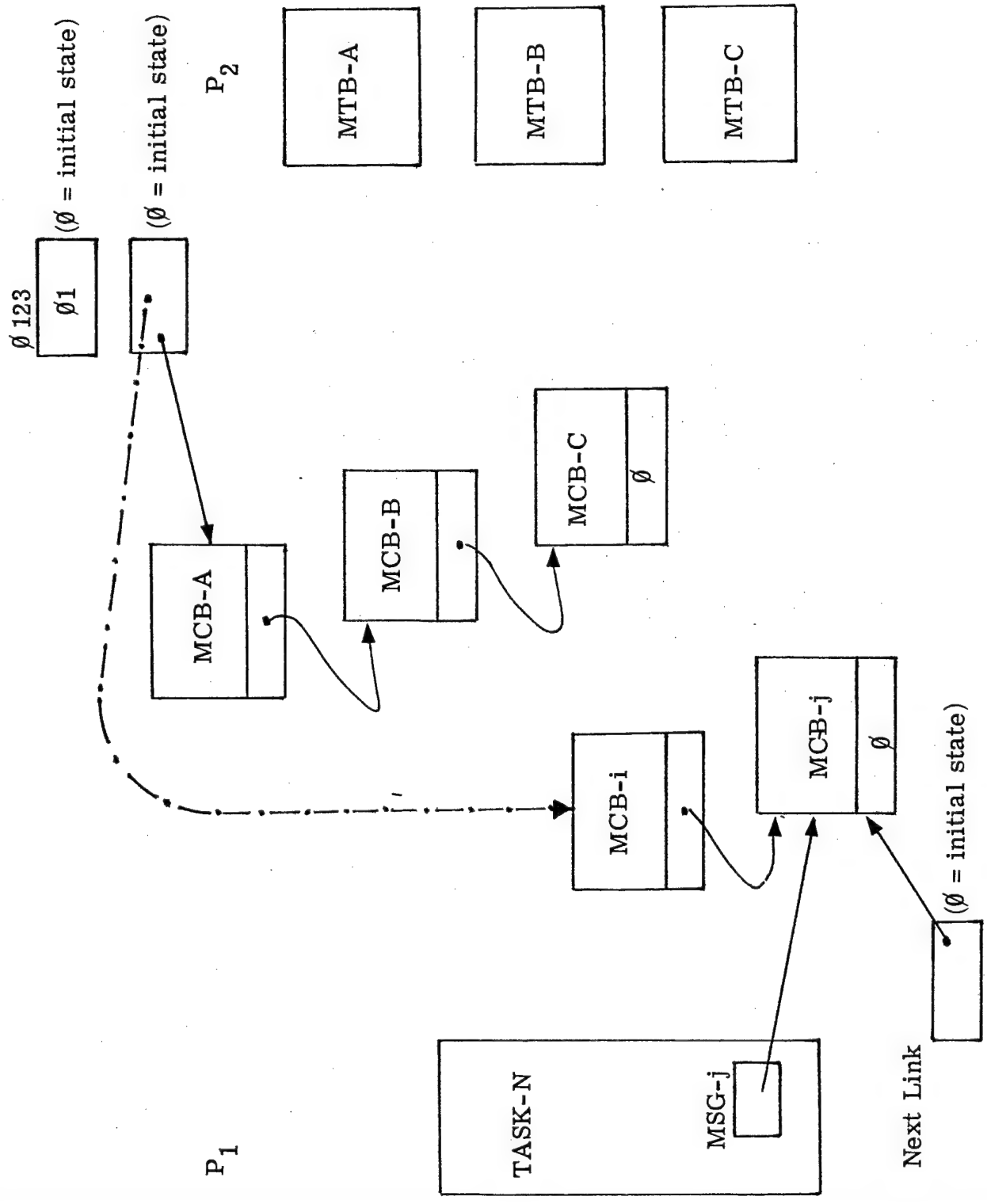



Figure 3a. Interprocessor Communication Example

3.1.6.3 MCB Management (GTMNC and FRMNC) - The memory blocks used for interprocessor communication (MCB's) shall be controlled by the Get Main Common (GTMNC) and Free Main Common (FRMNC) subroutines. The three sets of MCB's (RMP/CP, RMP/AP, and CP/AP) shall be linked by EXINT (RMP EXINT, RMP EXINT, and CP EXINT, respectively) with forward pointers to the next MCB to form a queue of available MCB's. GTMNC uses the start of queue pointer (RCSTR, RASTR, and CASTR, respectively) to remove MCB's from one of the three free queues. FRMNC uses the end of queue pointer (RCEND, RAEND, and CAEND, respectively) to return MCB's to one of the three free queues.

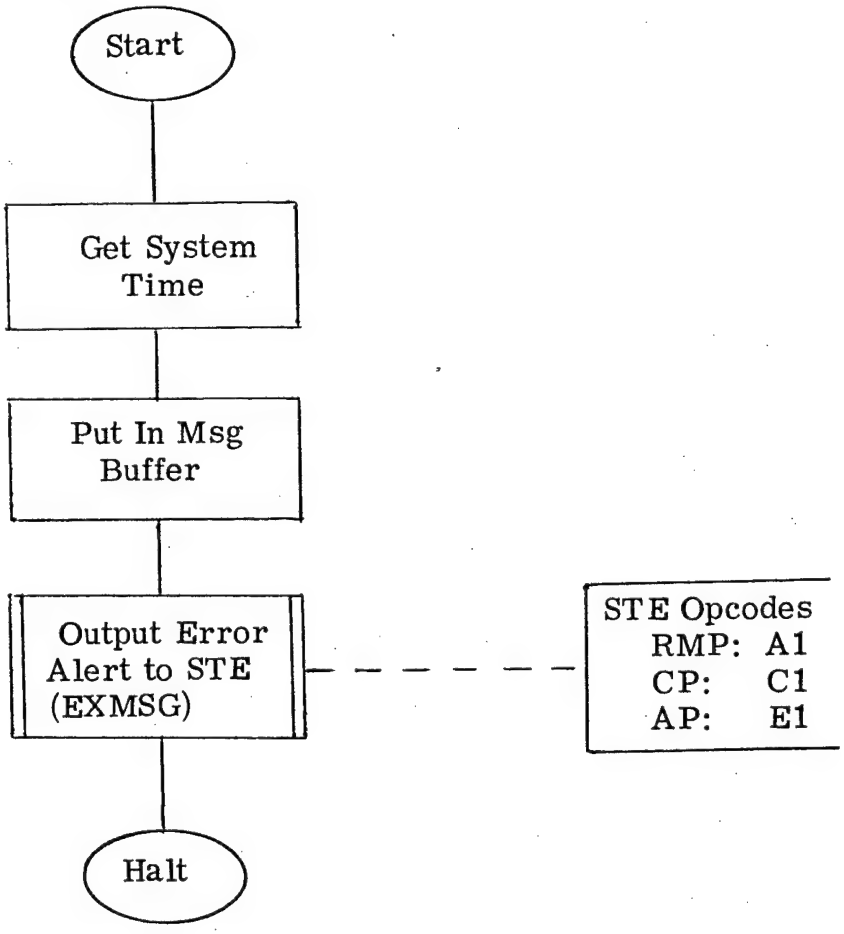
3.1.6.4 STE Message Sender (EXSTE) - EXSTE shall be an RMP Exec routine called by XMES and shall have the responsibility of sending Executive message type 5's to the STE. The messages shall be deposited into a circular buffer defined by the STE.

3.1.6.5 Signal Sorter Message Sender (EXSSS) - EXSSS shall be a CP Exec routine called by XMES and shall have the responsibility of sending Executive message type 19's to the Signal Sorter. The messages shall be deposited in either the high priority or low priority to - Sorter message buffers, depending on the message contents. If the message is sent via the high priority buffer, the CP shall interrupt the Sorter when the message is to be transmitted.

## 3.2 SUBPROGRAM FLOW DIAGRAMS

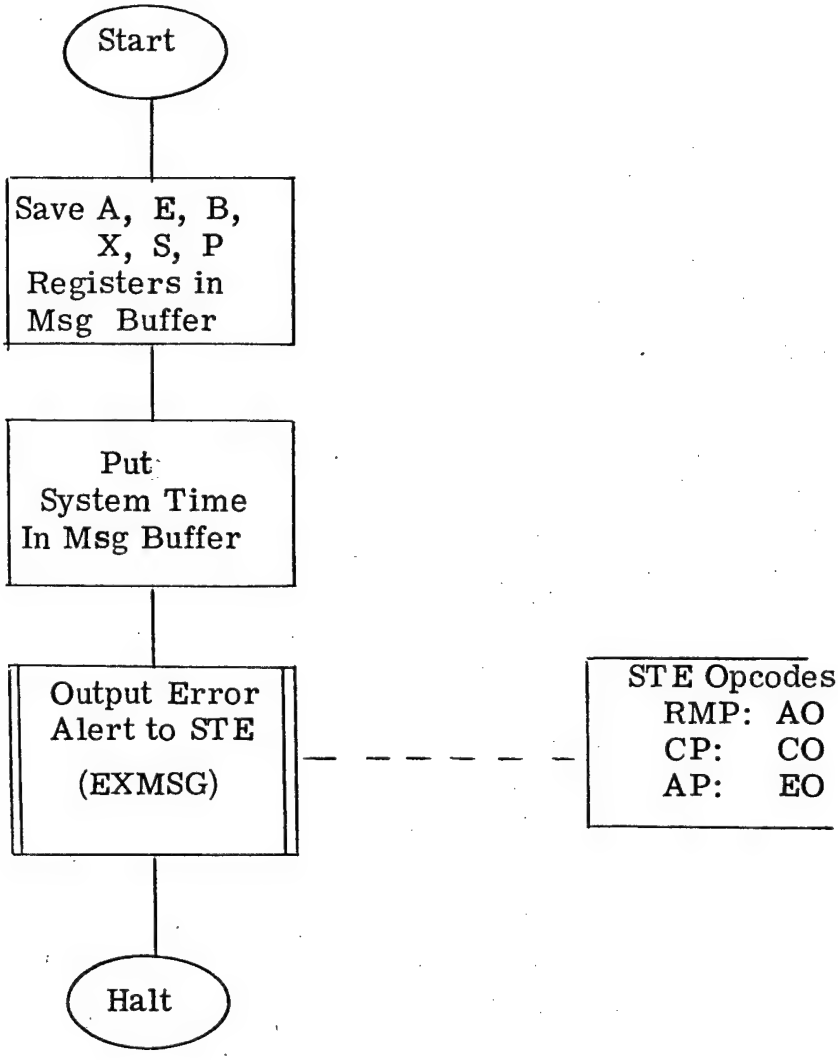
The following figures are the subprogram flow diagrams for the IEWS SC Executive function.

NOTUSD



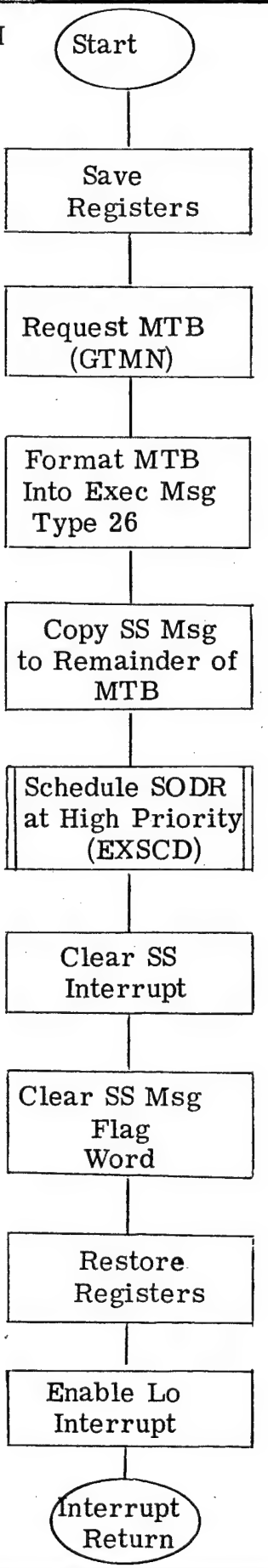

Routine: Unexpected Interrupt Handler
Processors: RMP, CP, AP

EXHNG




Routine: Bus Hung Interrupt Handler
Processors: RMP, CP, AP

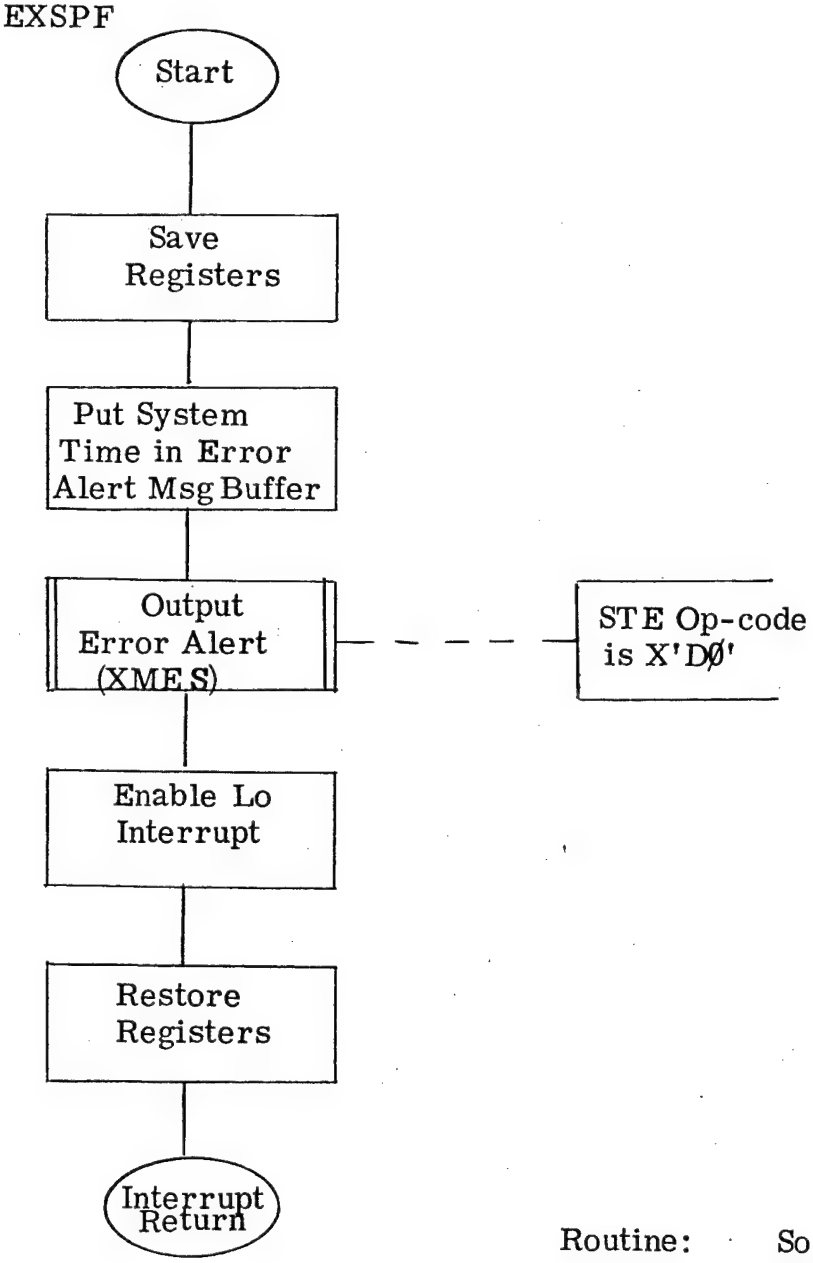
EXIH




Routine: Signal Sorter
High Priority Message
Interrupt Handler

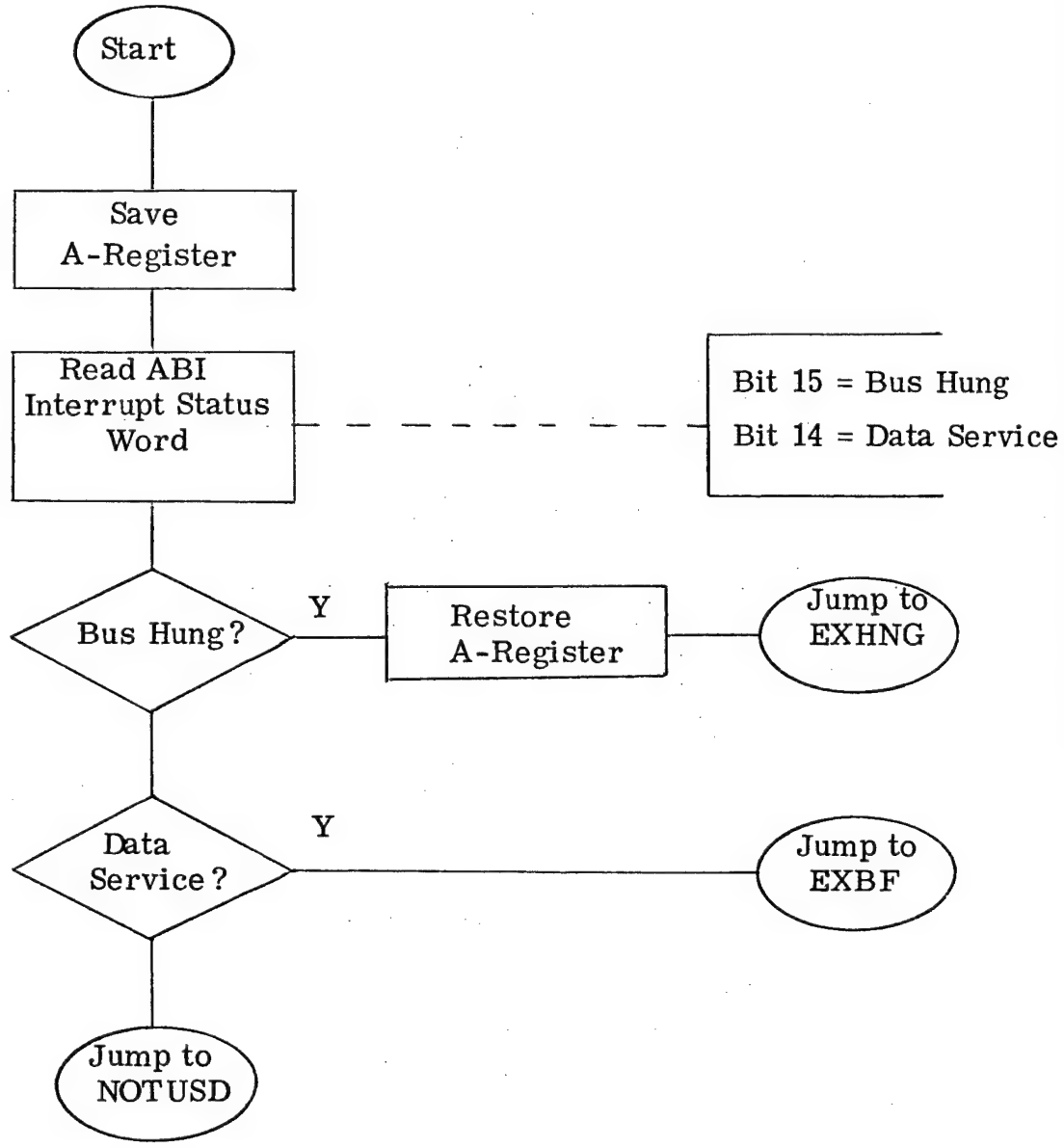
Processor: CP

EXSPF




Routine: Sorter Power Fail Interrupt Handler
Processor: CP

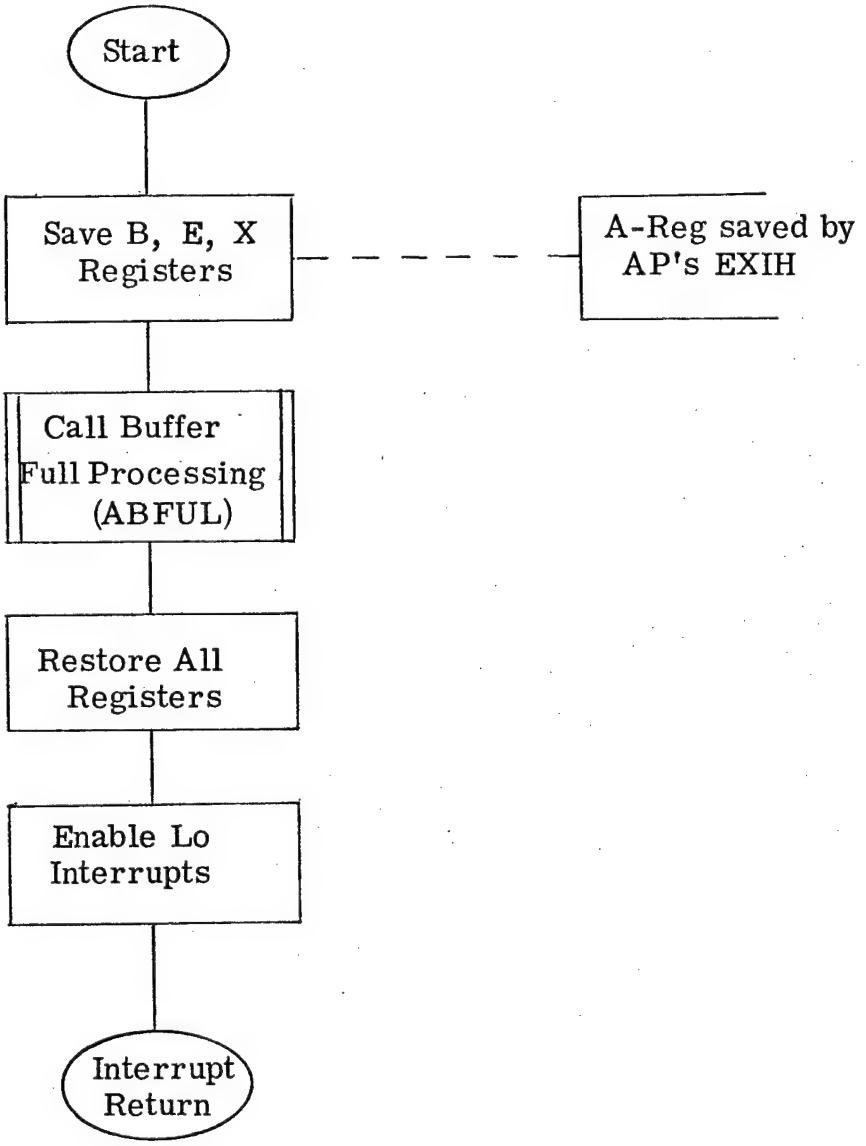
EXIH




Routine: Low Level Interrupt Handler

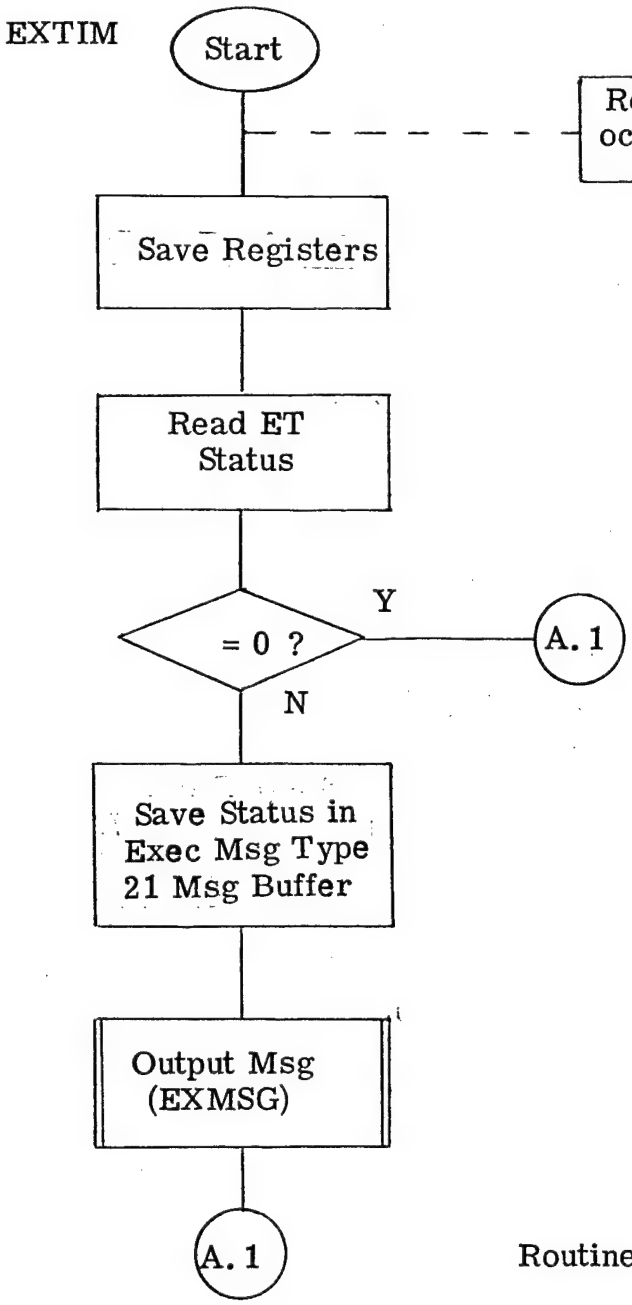
Processor: AP

EXBF



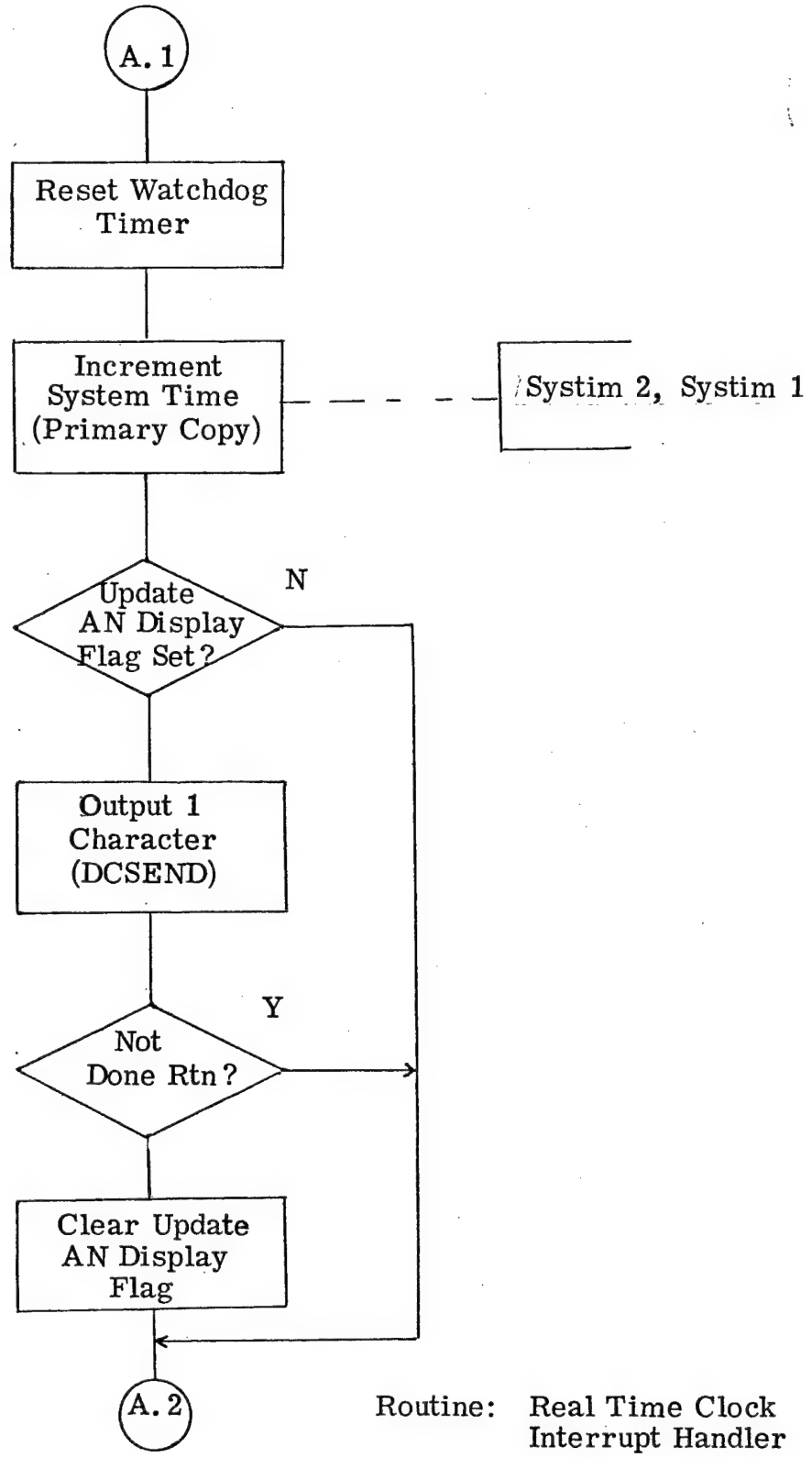

Routine: ABI Data Buffer Full Interrupt Handler

Processors: AP



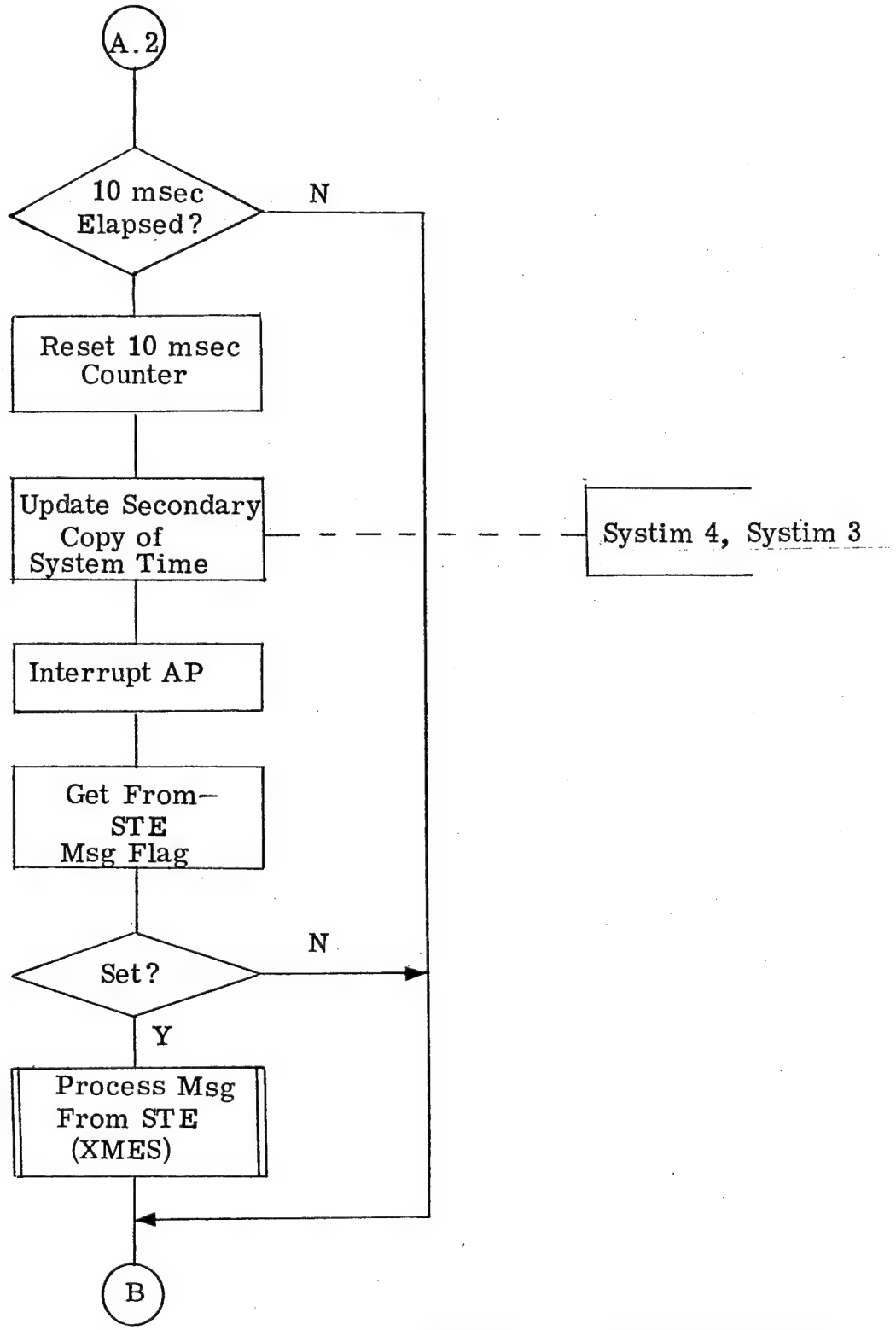

Routine: Real Time Clock Interrupt Handler

Processor: RMP



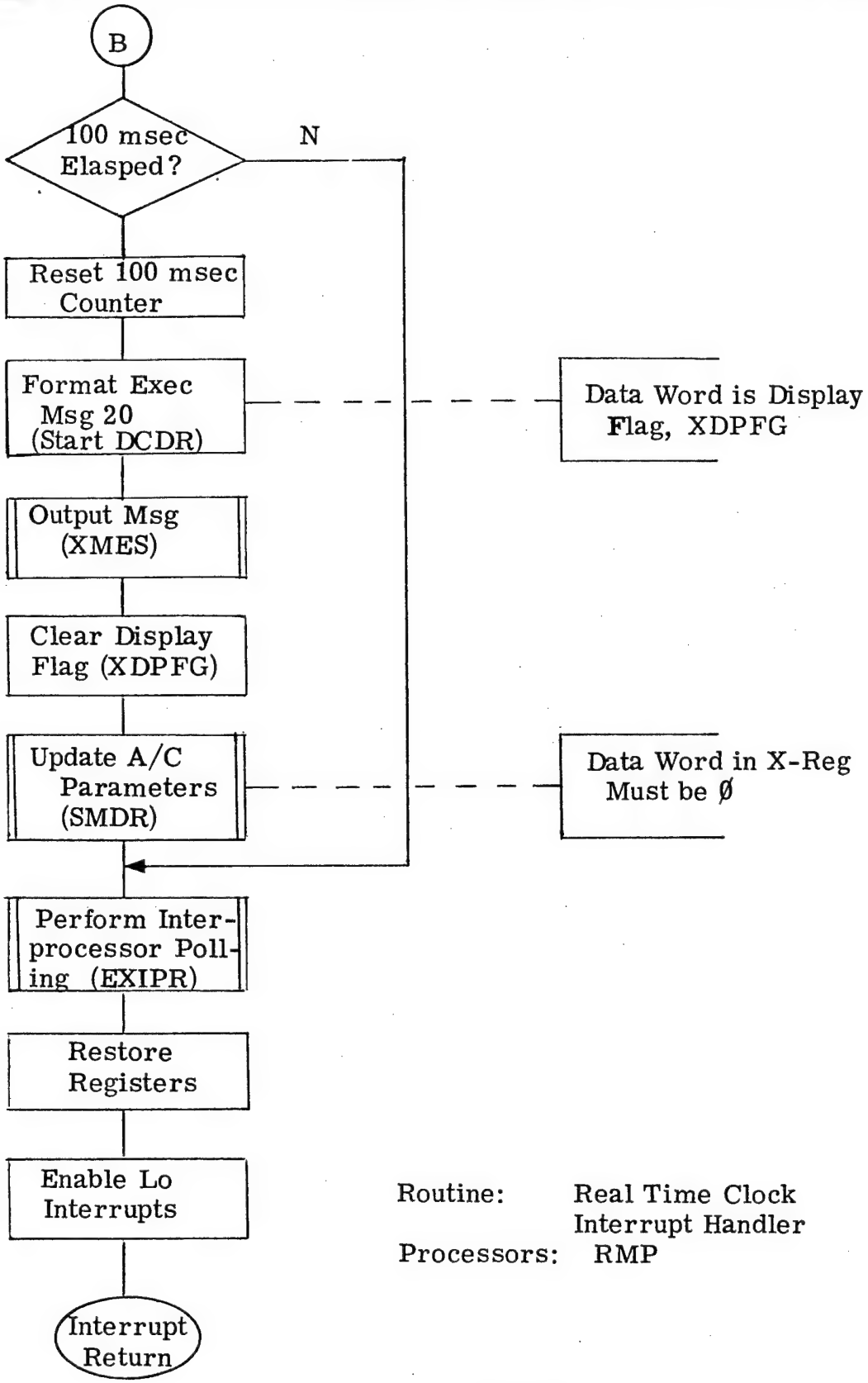

Routine: Real Time Clock Interrupt Handler

Processors: RMP

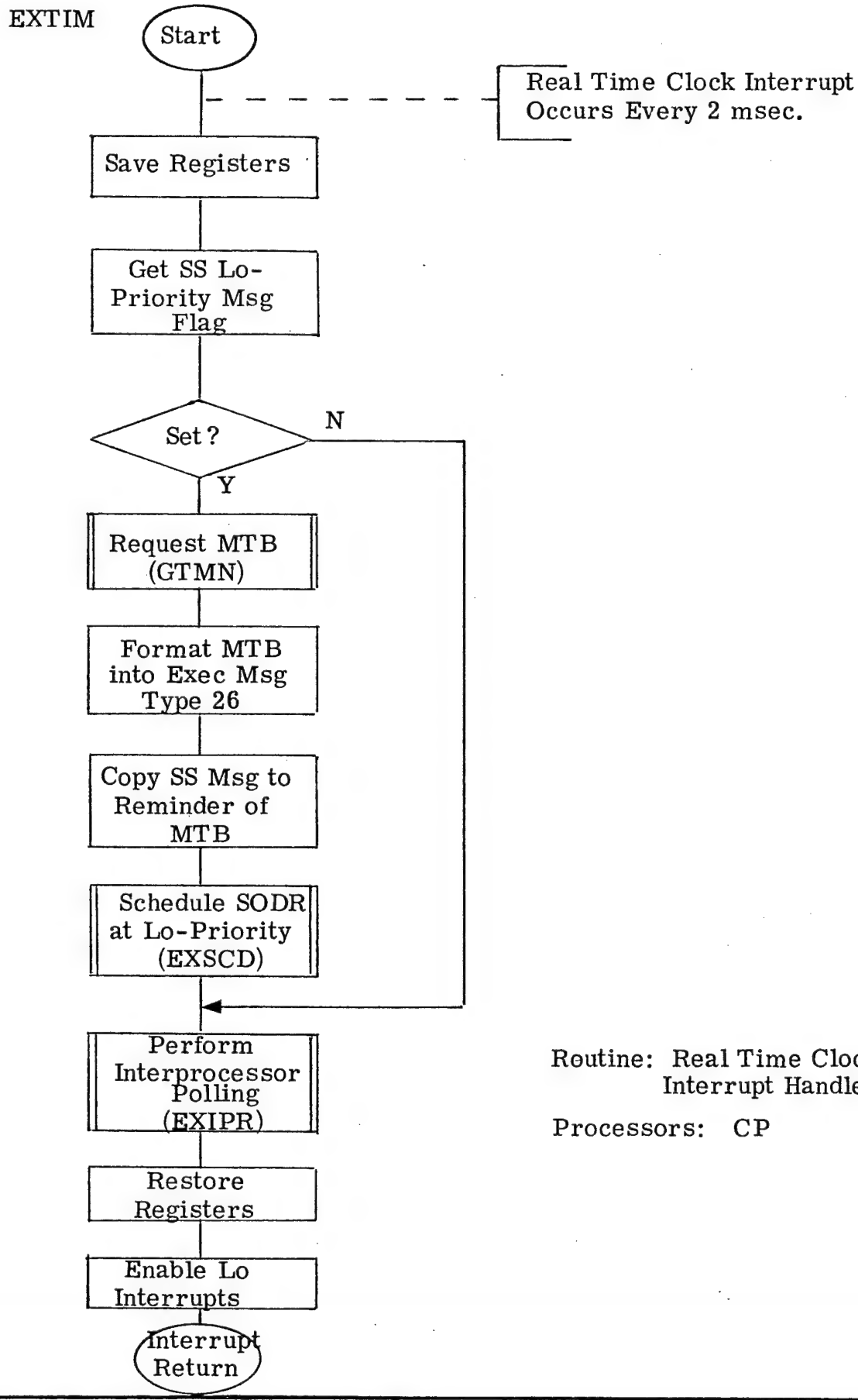



Routine: Real Time Clock Interrupt Handler

Processors: RMP



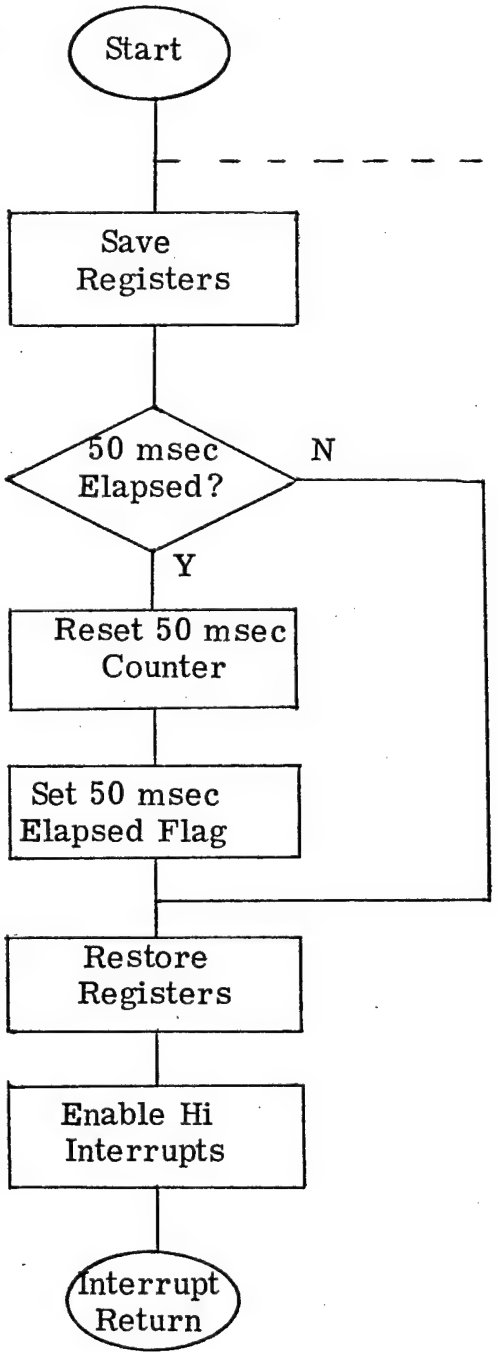

Routine: Real Time Clock Interrupt Handler
Processors: RMP




Routine: Real Time Clock Interrupt Handler

Processors: CP

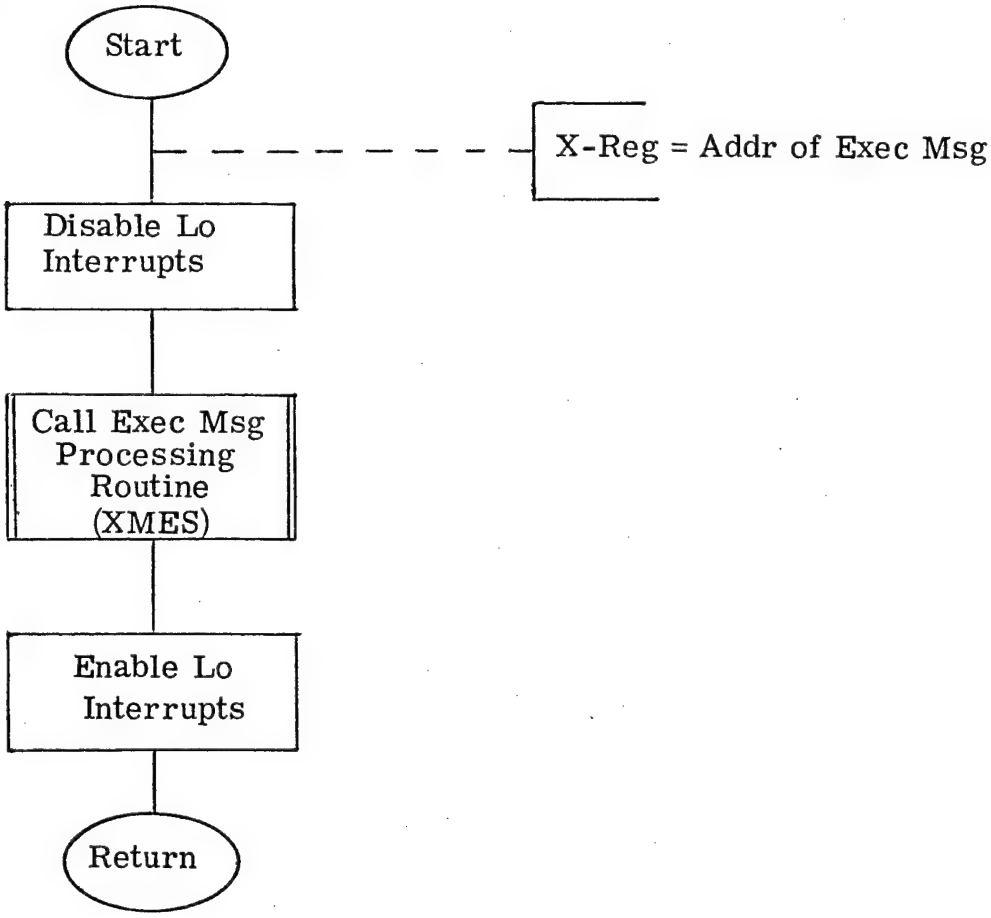
EXTIM




**Routine:** Simulated Real Time Clock Interrupt Handler (RMP Interrupt)

**Processors:** AP

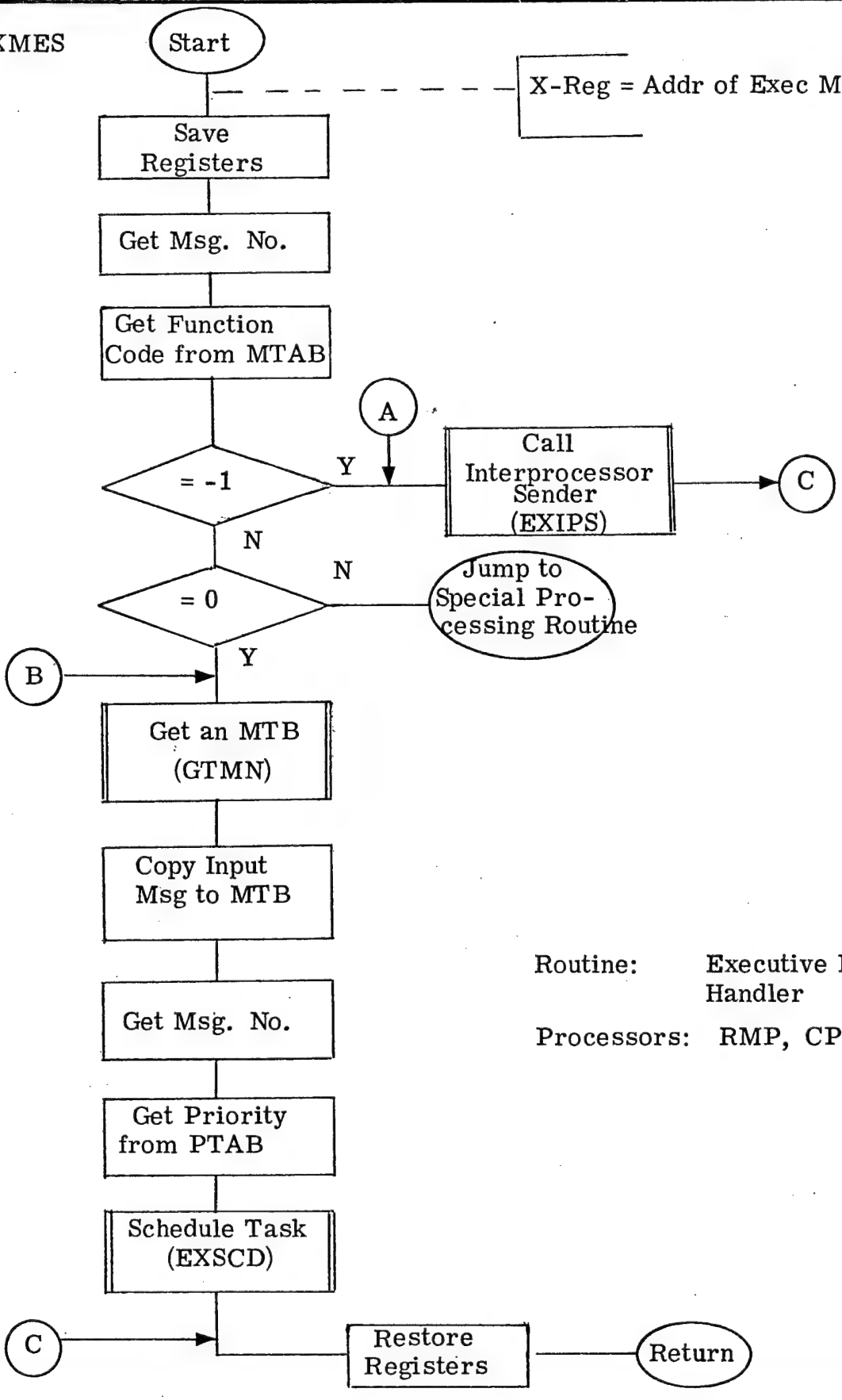
EXMES




Routine: Executive Message Handler

Processors: RMP, CP, AP

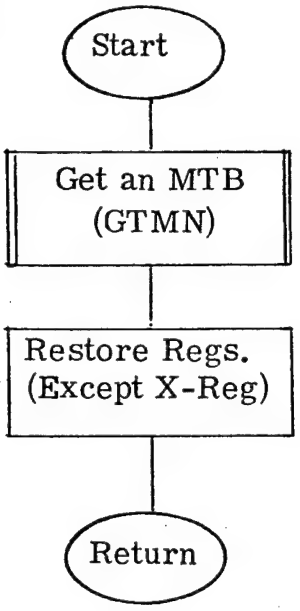
XMES



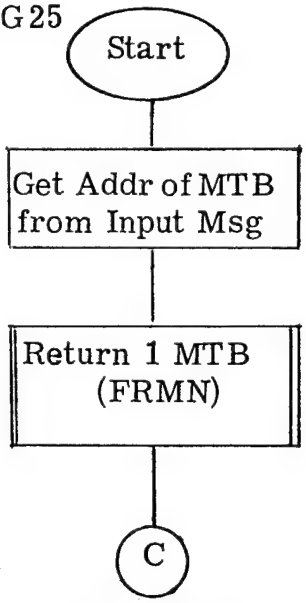

Routine: Executive Message Handler

Processors: RMP, CP, AP

MSG 24

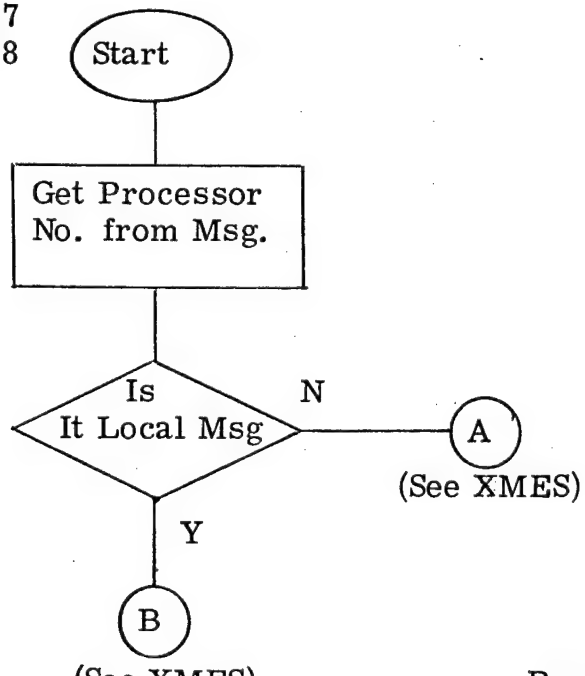



MSG 25




(See XMES)

MSG 27
MSG 28



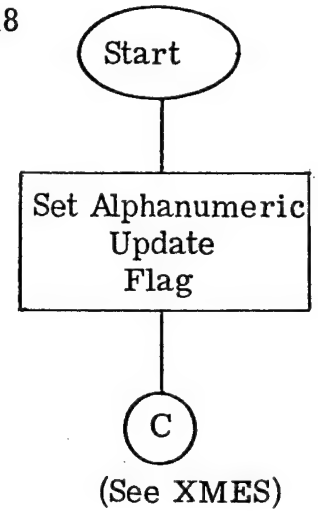

(See XMES)

(See XMES)

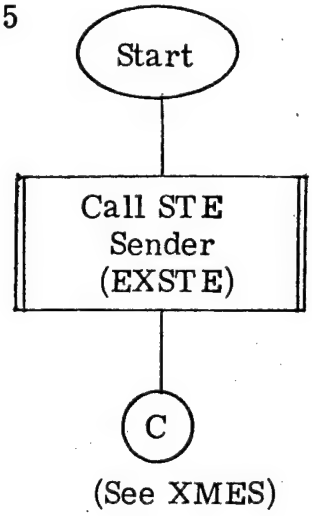
Routine: Executive Message Handler Special Processing

Processors: RMP, CP, AP

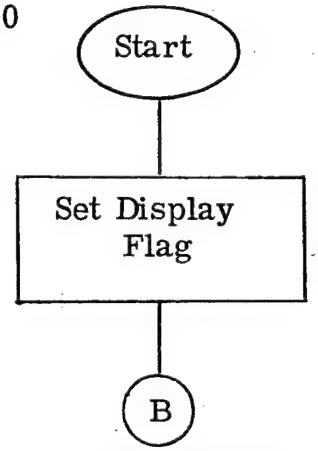
MSG 18




MSG 5



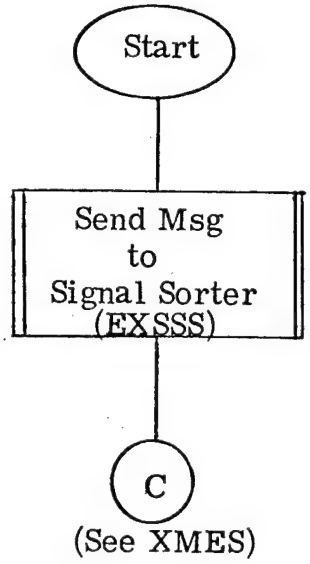

MSG 20




Routine: Executive Message Handler Special Processing

Processors: RMP

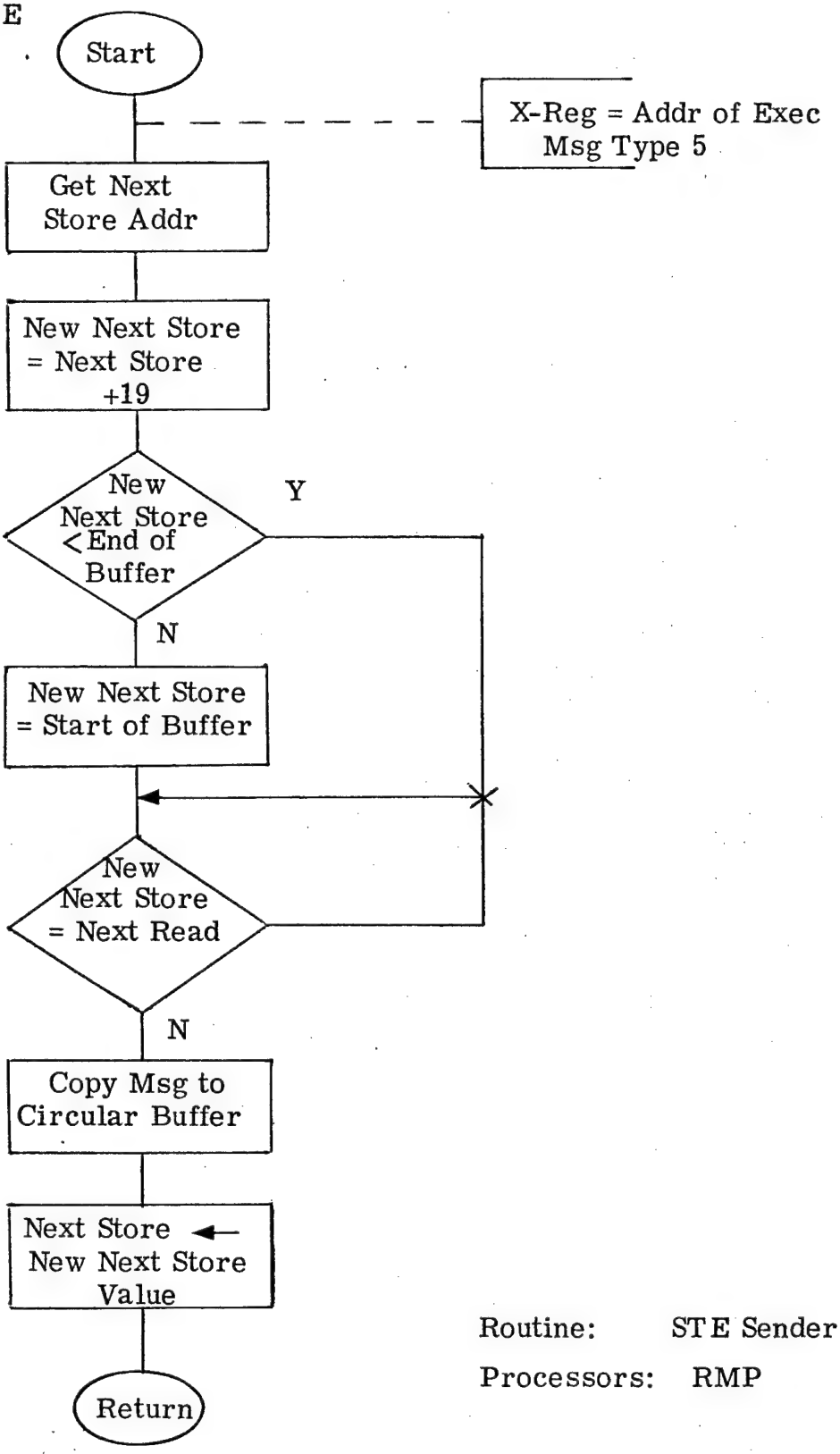
MSG 19




Routine: Executive Message Handler Special Processing

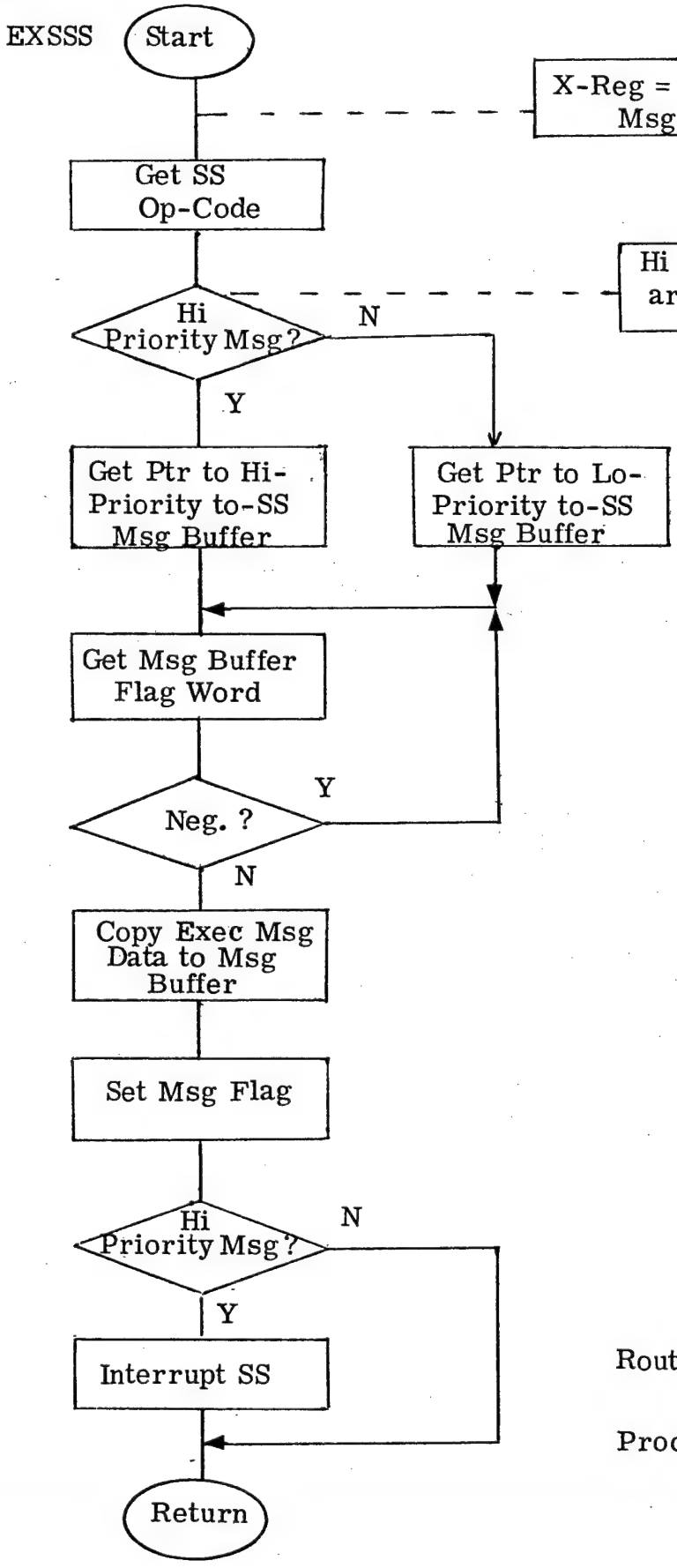
Processors: CP

EXSTE



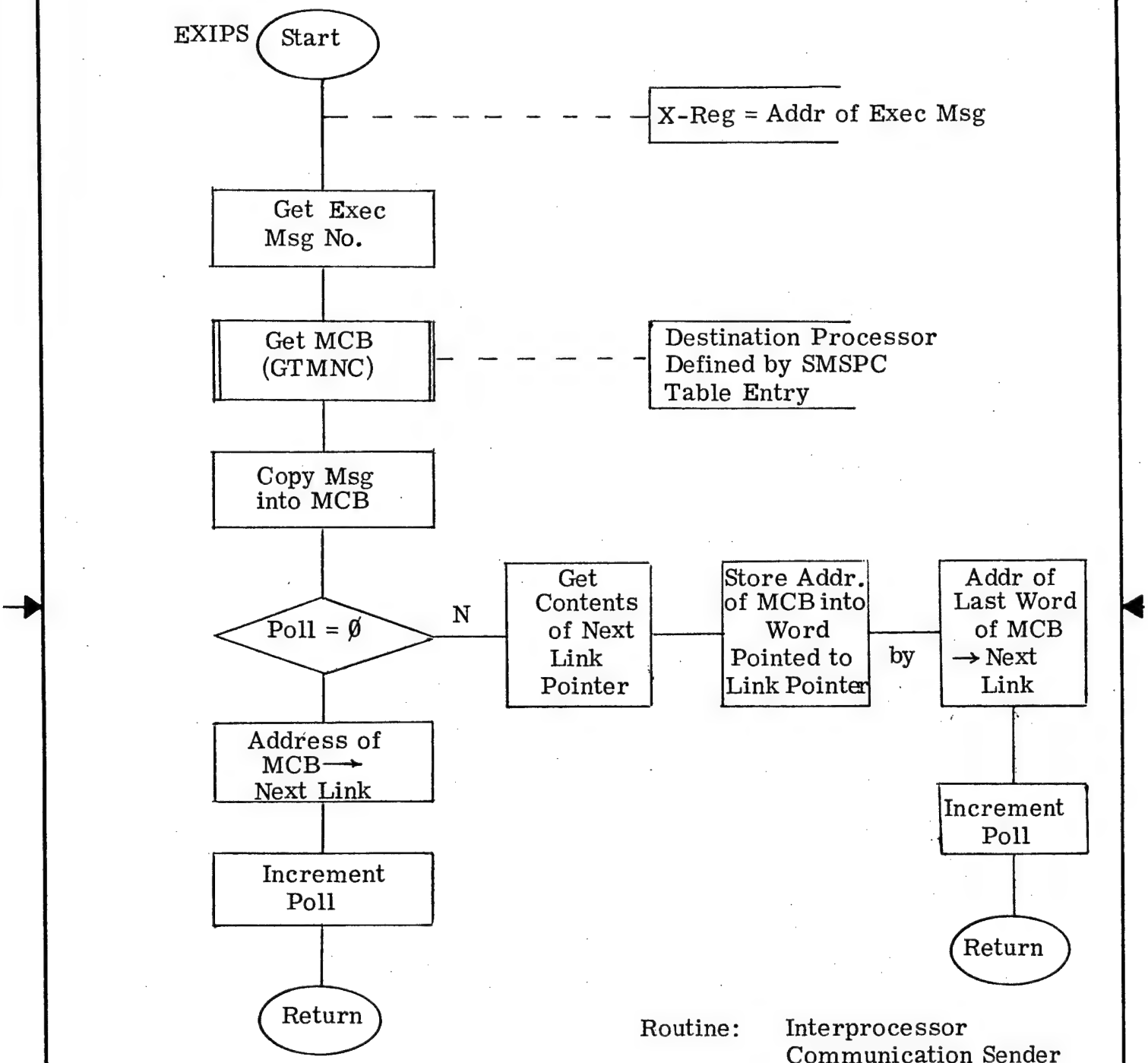

Routine: STE Sender

Processors: RMP




Routine: Signal Sorter Sender

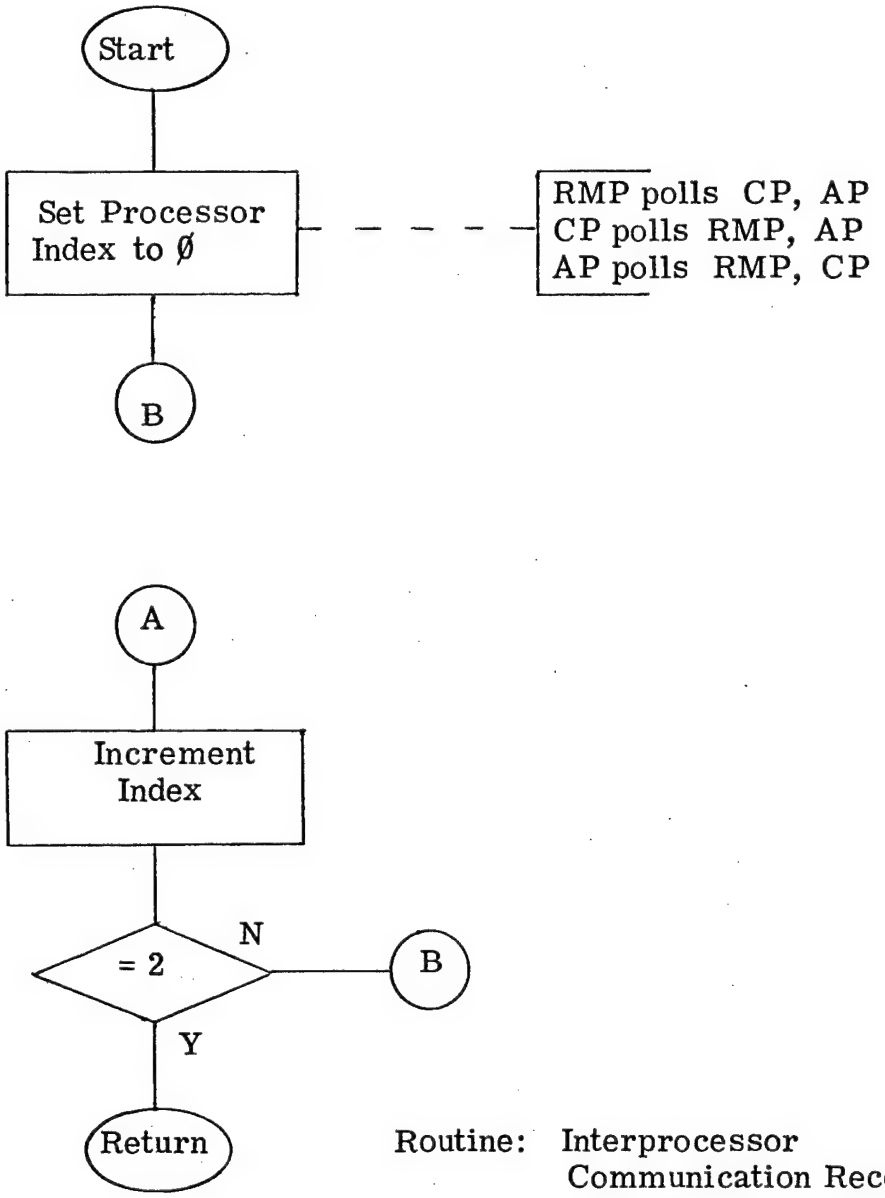
Processors: CP




Routine: Interprocessor Communication Sender

Processors: RMP, CP, AP

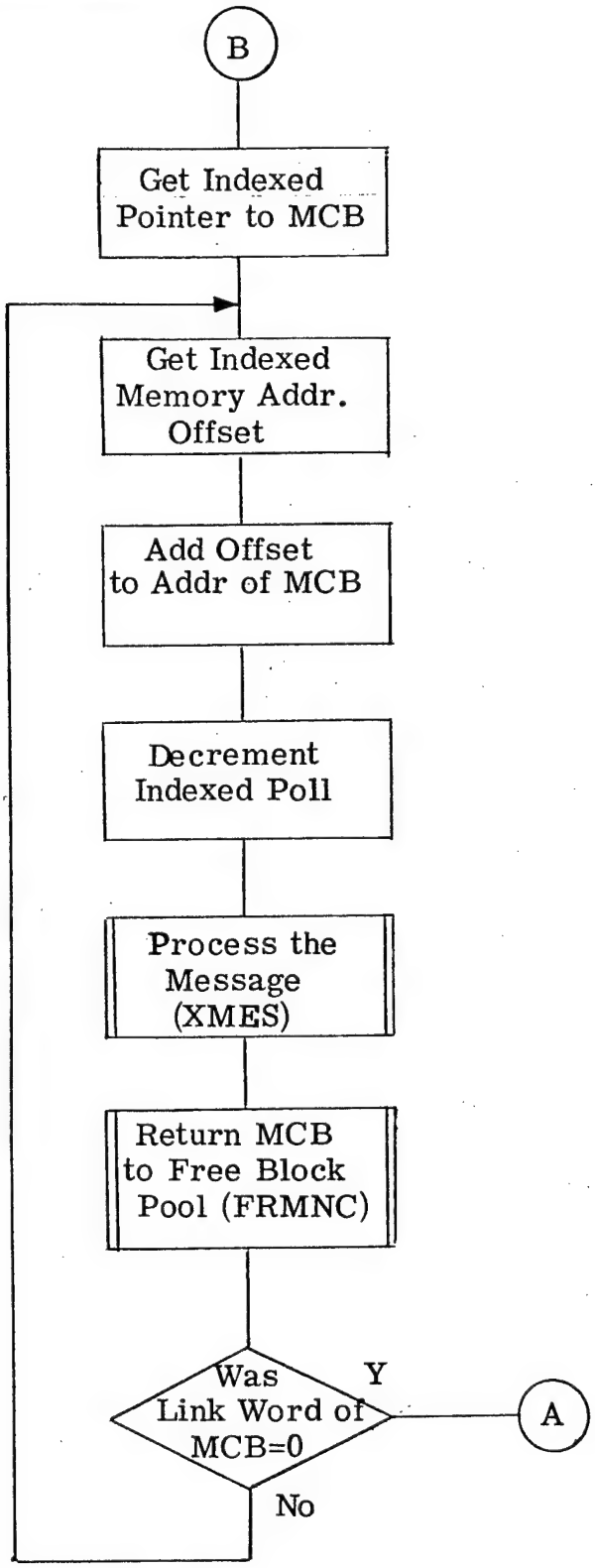
EXIPR




Routine: Interprocessor Communication Receiver

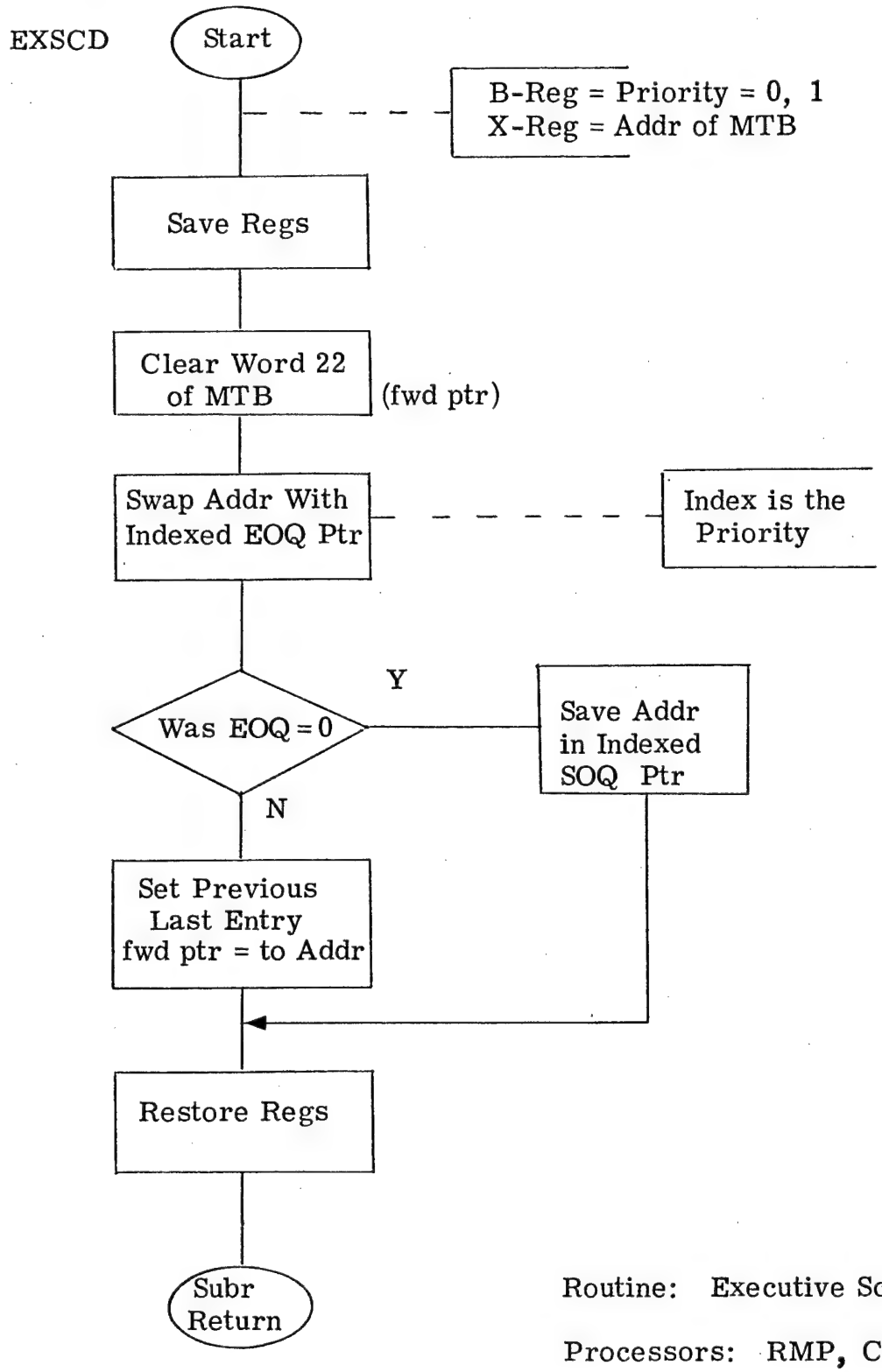
Processors: RMP, CP, AP

(1 of 2)



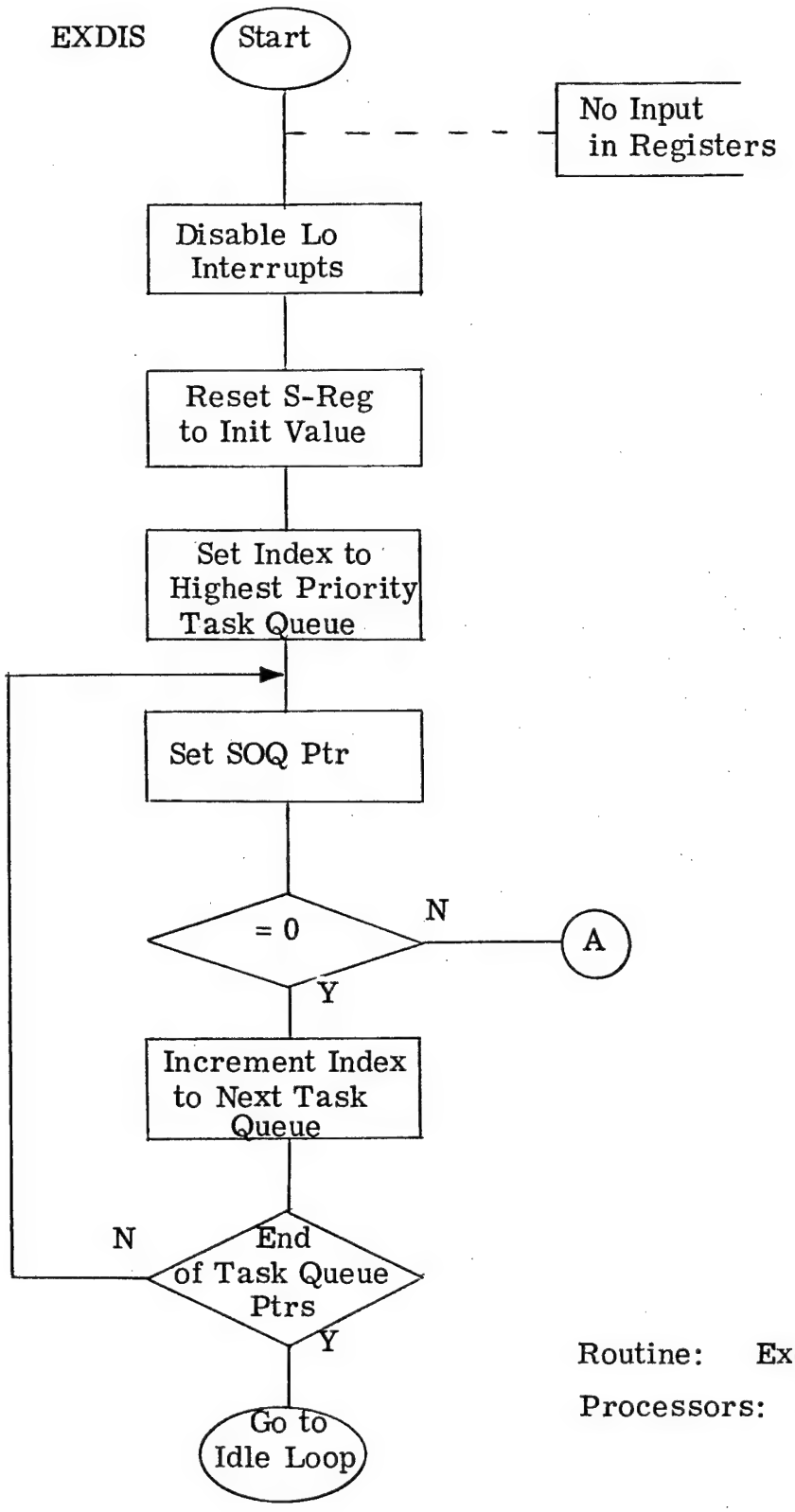

Routine: Interprocessor Communication Receiver

Processors: RMP, CP, AP



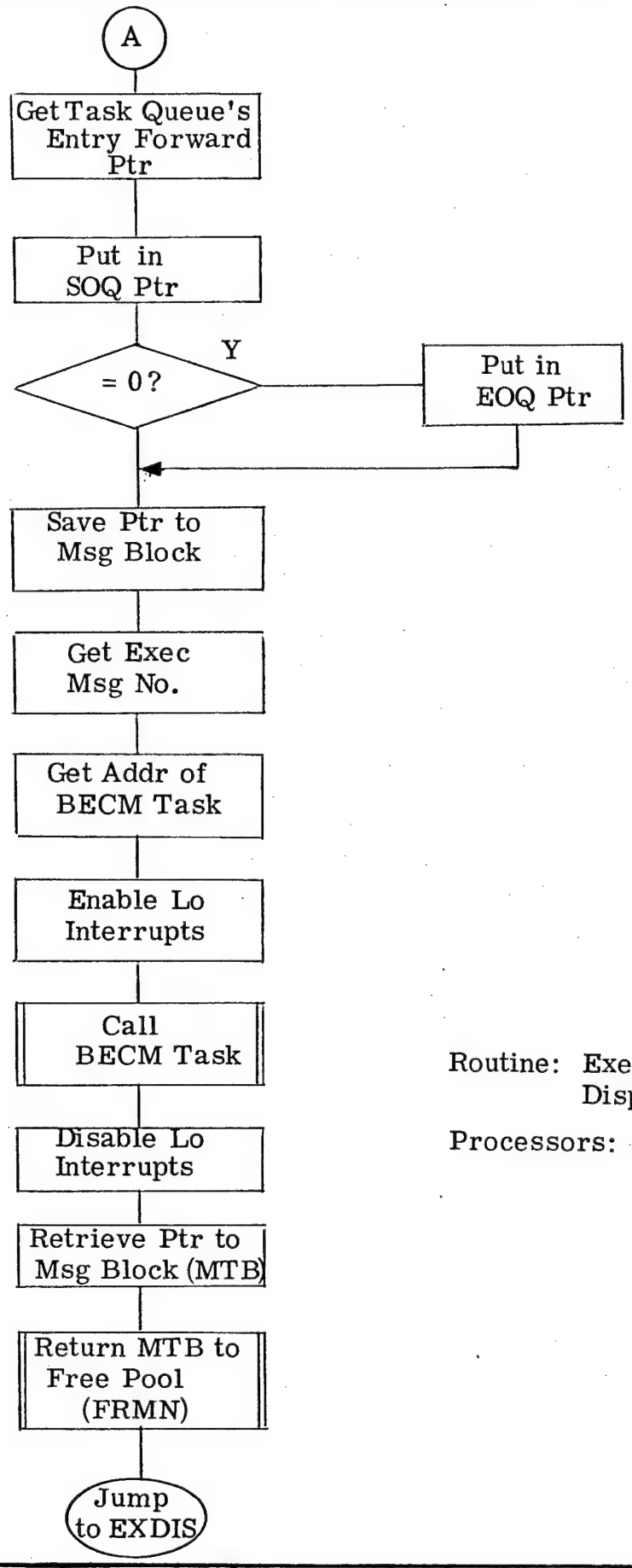

Routine: Executive Scheduler

Processors: RMP, CP, AP



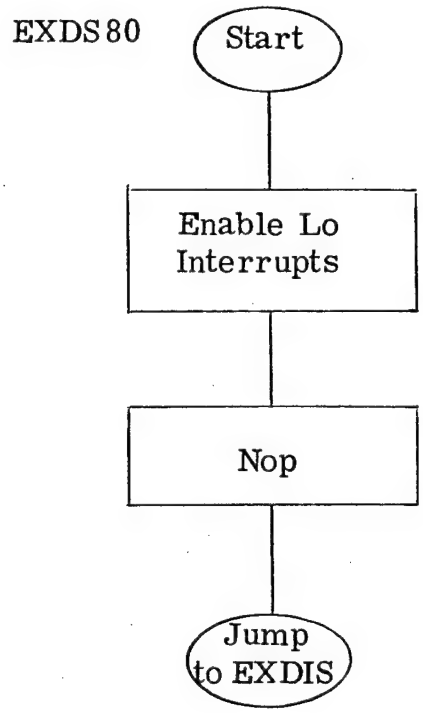

Routine: Executive Dispatcher

Processors: RMP, CP



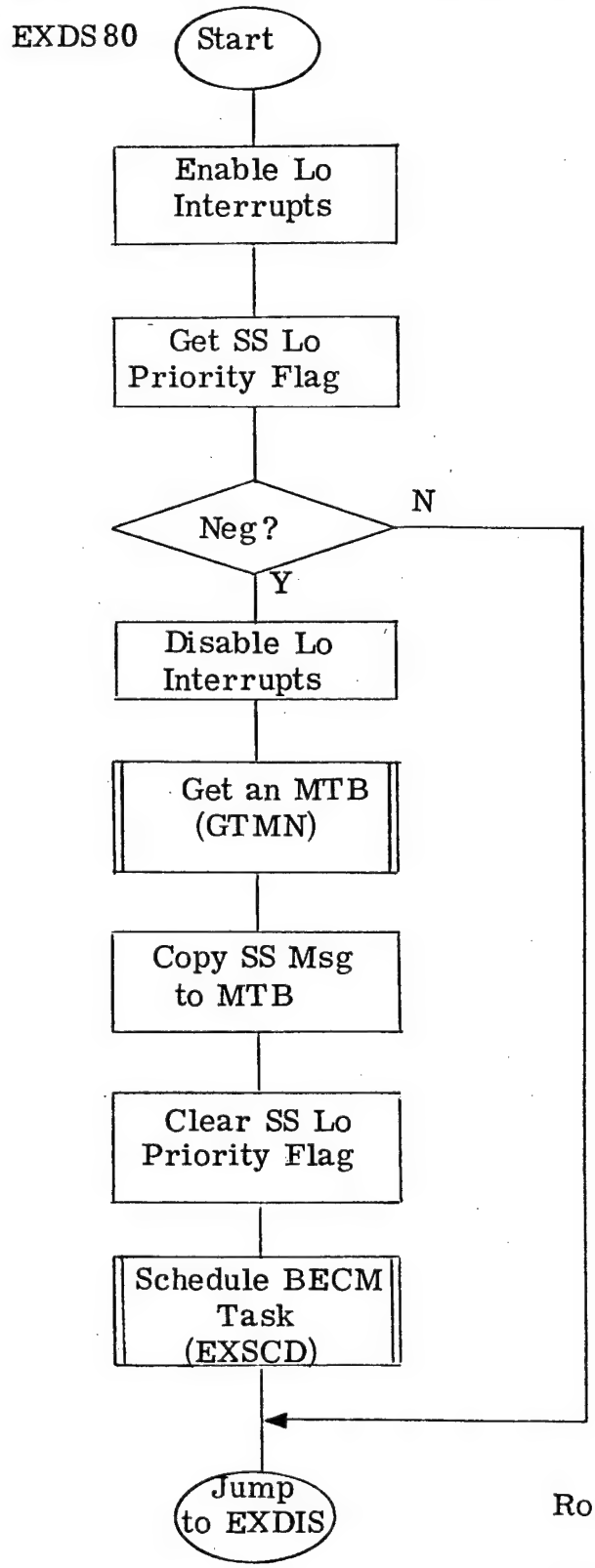

Routine: Executive Dispatcher

Processors: RMP, CP



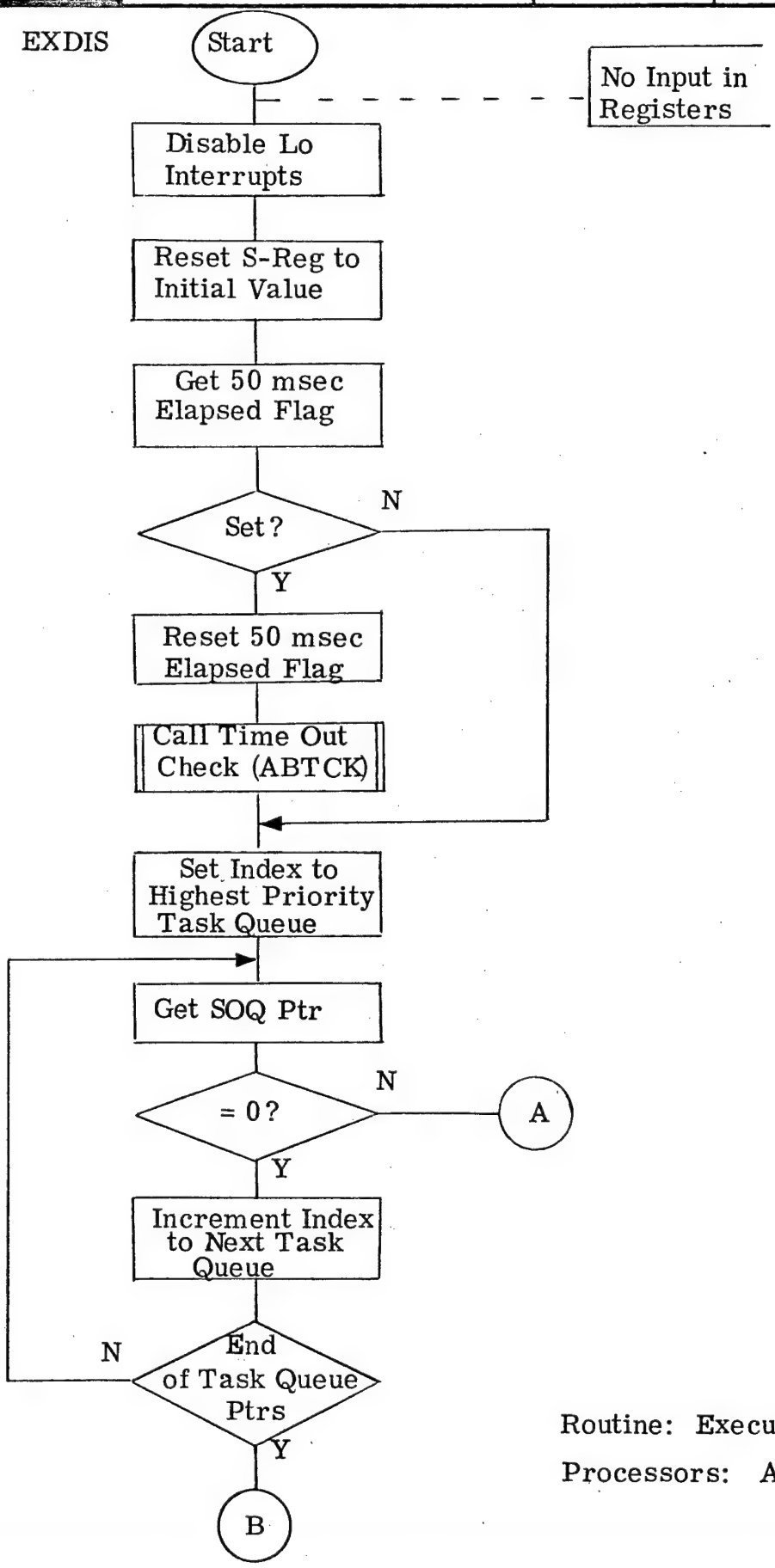

Routine: Executive Dispatcher Idle Loop

Processors: RMP




Routine: Executive Dispatcher Idle Loop

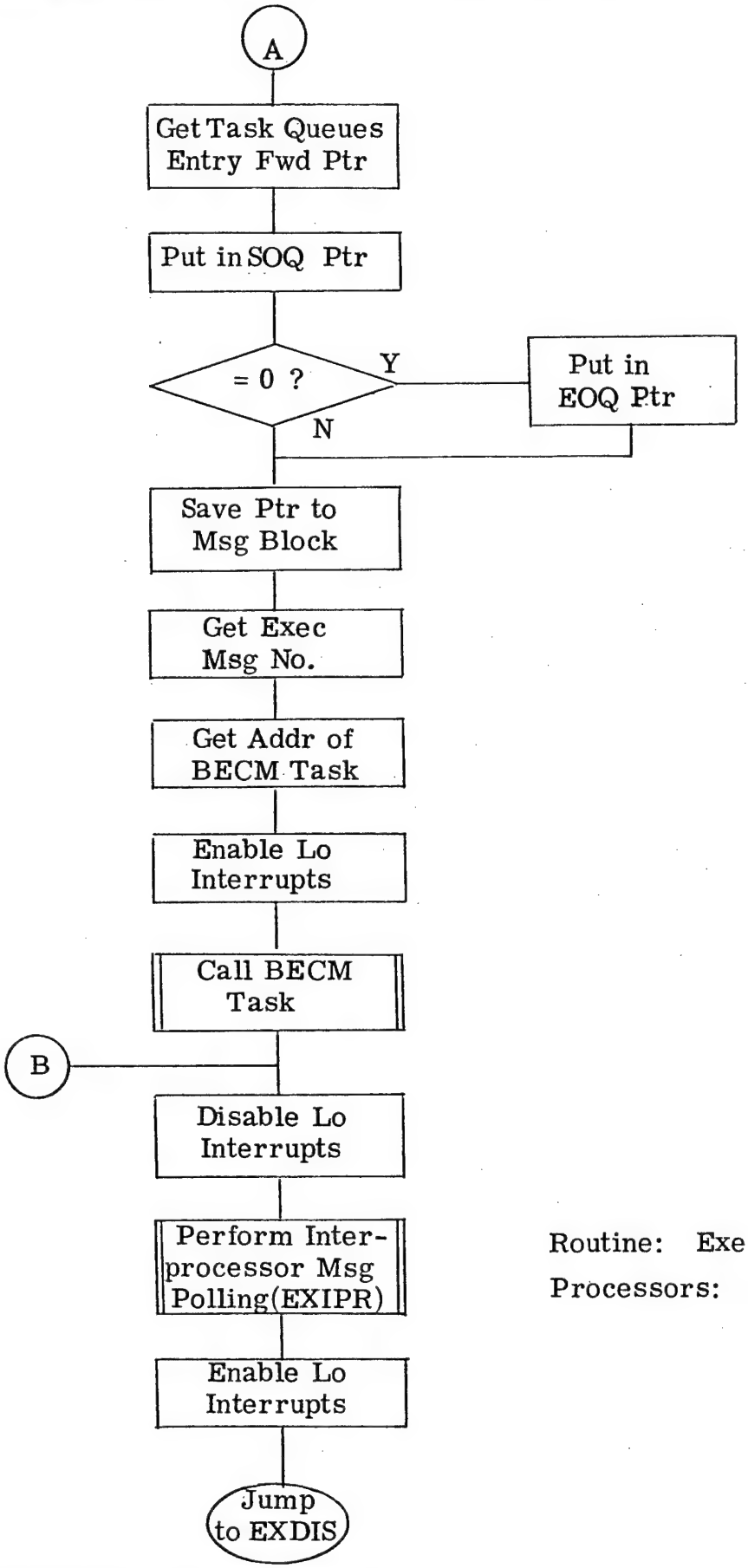
Processors: CP




Routine: Executive Dispatcher

Processors: AP

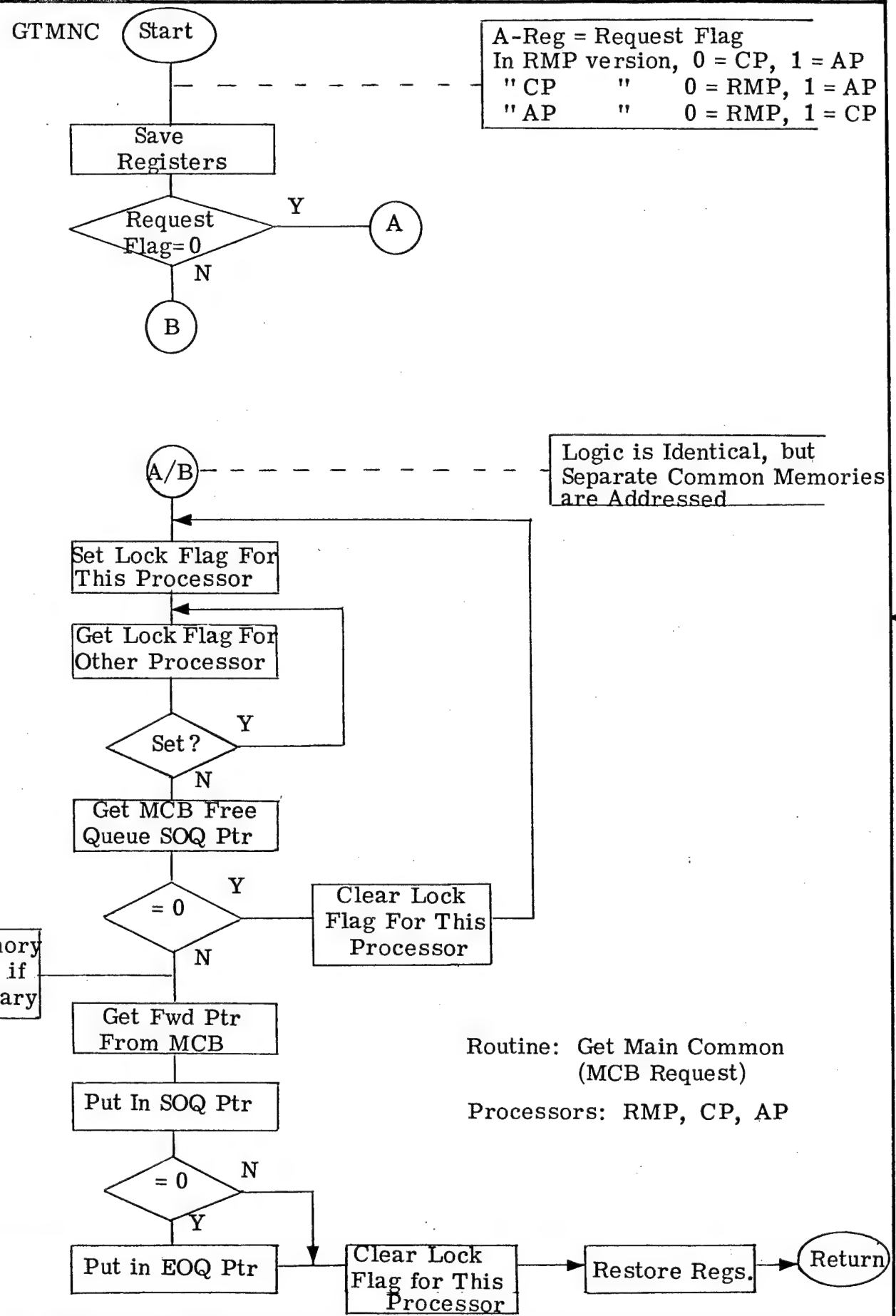
(1 of 2)

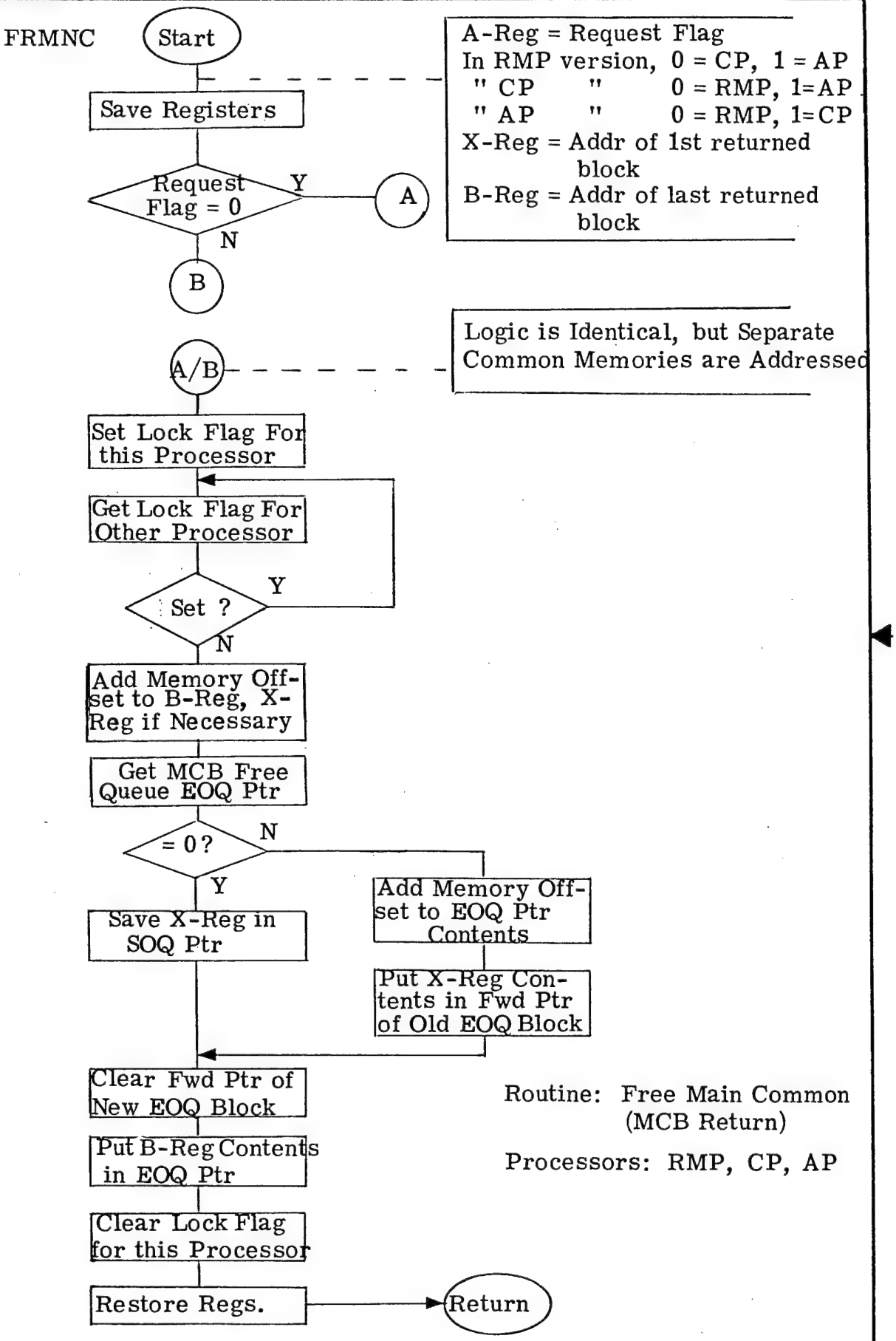



Routine: Executive Dispatcher

Processors: AP

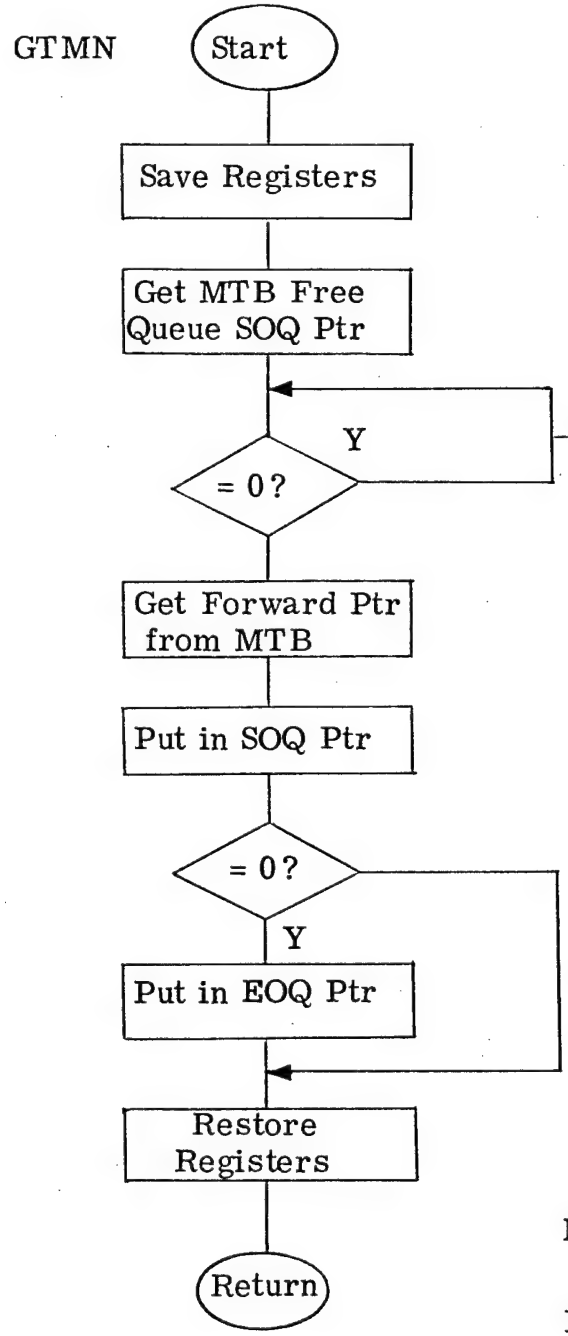
(2 of 2)


GTMNC
Start
A-Reg = Request Flag
In RMP version, 0 = CP, 1 = AP
" CP " 0 = RMP, 1 = AP
" AP " 0 = RMP, 1 = CP
Save Registers
Request Flag=0
Y
N
A
B
A/B
Logic is Identical, but Separate Common Memories are Addressed
Set Lock Flag For This Processor
Get Lock Flag For Other Processor
Set?
Y
N
Get MCB Free Queue SOQ Ptr
= 0
Y
N
Clear Lock Flag For This Processor
Add Memory Offset, if Necessary
Get Fwd Ptr From MCB
Put In SOQ Ptr
= 0
N
Y
Put in EOQ Ptr
Clear Lock Flag for This Processor
Restore Regs.
Return
Routine: Get Main Common (MCB Request)
Processors: RMP, CP, AP



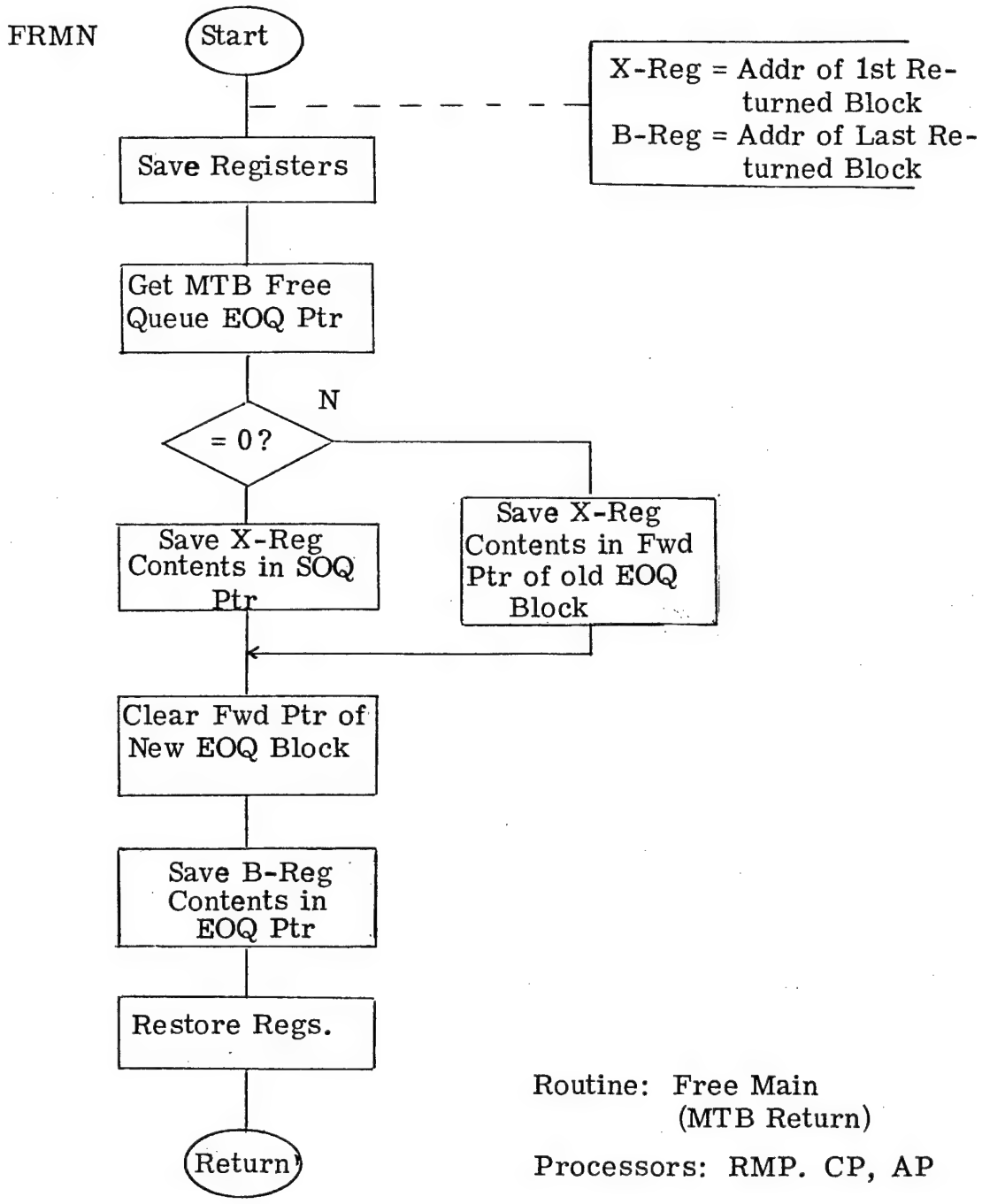

Routine: Free Main Common (MCB Return)

Processors: RMP, CP, AP




Routine: Get Main
(MTB Request)

Processors: RMP, CP, AP




Routine: Free Main (MTB Return)

Processors: RMP. CP, AP

## 3.3 COMPUTER SUBPROGRAM ENVIRONMENT

### 3.3.1 Tables

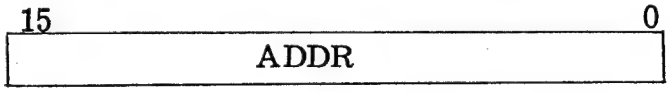
#### 3.3.1.1 Executive Task Table -

a) Table Name: XTASK

b) Purpose & Type: XTASK shall be used to determine the address of a background ECM processing task when that task is to be dispatched. XTASK is a fixed length table.

c) Size & Indexing Procedure: 32 entries of one 16-Bit word. All entries shall be referenced by indexed displacement from the start of the table. Index shall be executive message number.

d) Entry Format:




| Field | Description | Units | LSB |
|---|---|---|---|
| ADDR | BECM Task Address | N/A | N/A |

#### 3.3.1.2 Message Table -

a) Table Name: MTAB

b) Purpose & Type: MTAB shall be used by XMES to identify an executive message when it is received by the Executive (received from BECM processing or interprocessor communication) MTAB is a fixed length table.

c) Size & Indexing Procedure: 32 entries of one 16-Bit word. All entries shall be referenced by indexed displacement from the start of the table. Index shall be executive message number.

d) Entry Format:

| 15 | 0 |
|---|---|
| Flag | |

| Field | Description | Units | LSB |
|---|---|---|---|
| Flag | Flag Word<br>0 means executive message is to be processed locally<br>-1 means message is to be relayed to another processor<br>Otherwise, flag is used as the address of a special processing routine for this message type. | N/A | N/A |

3.3.1.3 Message Priority Table -

a) Table Name: PTAB

b) Purpose & Type: PTAB shall be used by XMES to determine the priority of an executive message when it has been determined that the message is to be processed locally (i.e., in this processor). PTAB is a fixed length table.

c) Size & Indexing Procedure: 32 entries of one 16-Bit word. All entries shall be referenced by indexed displacement from the start of the table. Index shall be executive message number.

d) Entry Format

| 15 | 0 |
|---|---|
| Priority | |

| Field | Description | Units | LSB |
|---|---|---|---|
| Priority | Executive message priority for a given processor. IEWS, ADM has 2 priorities:<br>0 = high<br>1 = low | N/A | N/A |

3.3.2 Variables

None.

3.3.3 Constants

None.

3.3.4 Flags

None.

3.3.5 Indices

None.

3.3.6 Common Data Base References

The RMP Initializer (EXINT) shall set the following common data base items to the initial value:

| | |
|---|---|
| AZ | Azimuth Link Table |
| JS | Jam Status File |
| RF | Resource File |
| CD | Display/Control Status File |
| PI | Polar Image File |
| AN | Alphanumeric Image File |

The CP Initializer shall set the following common data base items to the initial value:

| | |
|---|---|
| ATC | Amplitude Threshold Constant |
| ETF | Emitter Track File |

The AP Initializer shall call subroutine AB2IN to set common data base items to the initial value (see CSDD, ABI Management, IEWS, ADM, Document No. 53959-GT-0754).

3.3.7 Queues

3.3.7.1 Priority Task Queues

a) Purpose: - The task queues shall be used to hold the backlog of requests for ECM processing in a given processor. There shall be two queues: high priority and low priority. Initially, these queues shall be empty.

b) Queue Definition Variables: - Each of the two task queues shall be defined by a start of queue (SOQ) pointer (XSOQØ, XEOQØ for the high priority and XSOQ1 and XEOQ1 for the low priority). The SOQ pointer contains the address of the oldest entry in the queue or Ø if the queue is empty. The EOQ pointer contains the address of the most recent addition to the queue or Ø.

c) Entry Format: - See Figure 4.

3.3.7.2 Free MTB Queue

a) Purpose: - The Free MTB Queue shall be used to hold the surplus of memory blocks used for dynamic local storage allocation in a given processor. Inititally this queue shall contain all memory blocks available for use in a processor.

b) Queue Definition Variables: - SOQ pointer shall be XSTR. EOQ pointer shall be XEND. XBUF shall be the low memory address of the memory allocated for MTB's.

c) Entry Format: - See Figure 5.

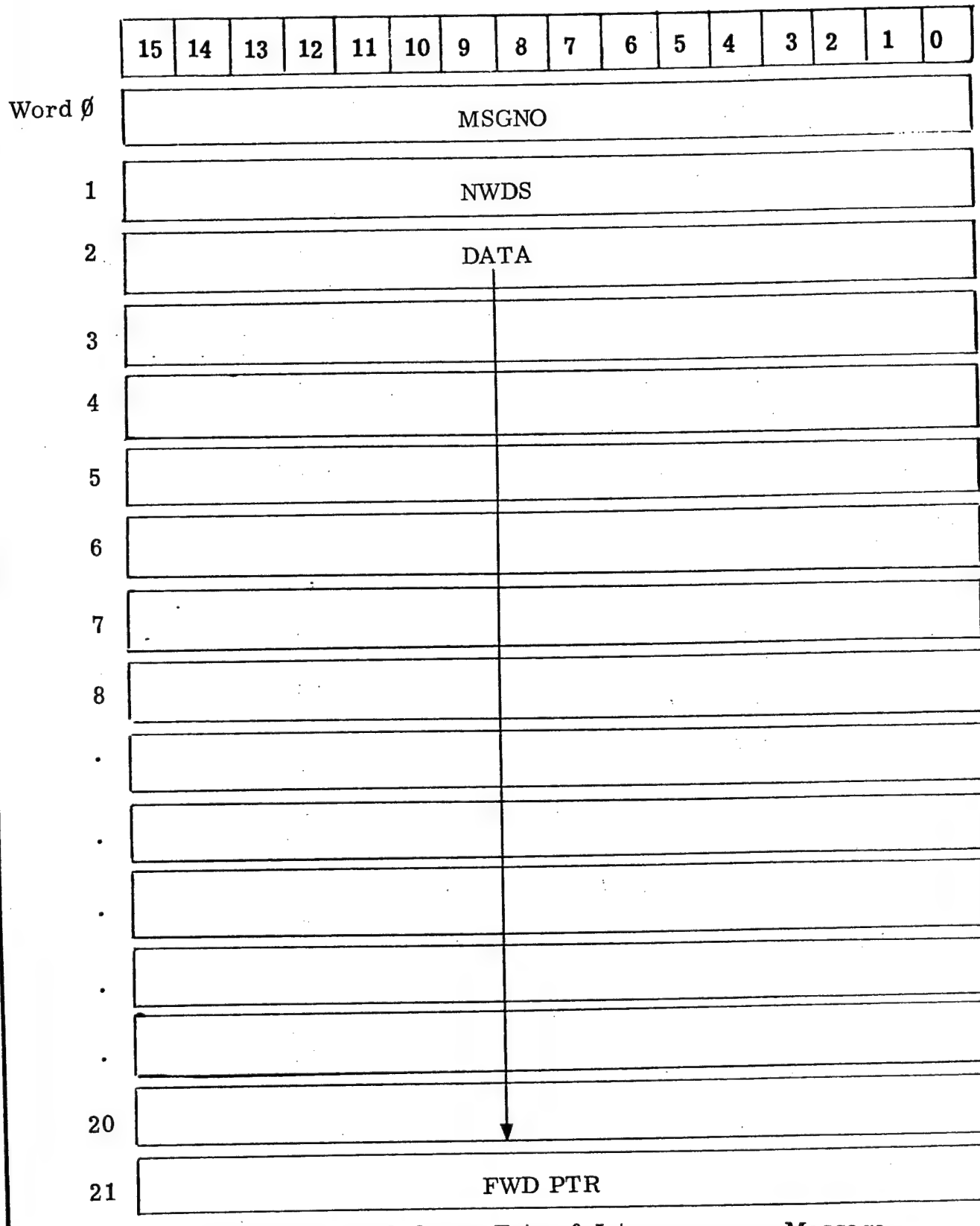



Figure 4a. Task Queue Entry & Interprocessor Message Transfer Queue Entry

| Field | Description | Units | LSB |
|---|---|---|---|
| MSGNO | Executive Message Number | N/A | 1 |
| NWDS | Number of Words of Data | N/A | 1 |
| DATA | Executive Message Data | N/A | N/A |
| FWDPTR | Address of Next Entry in Queue or Ø if this Entry is EOQ Entry | N/A | N/A |

Figure 4b. Task Queue Entry & Interprocessor Message Transfer Queue Entry

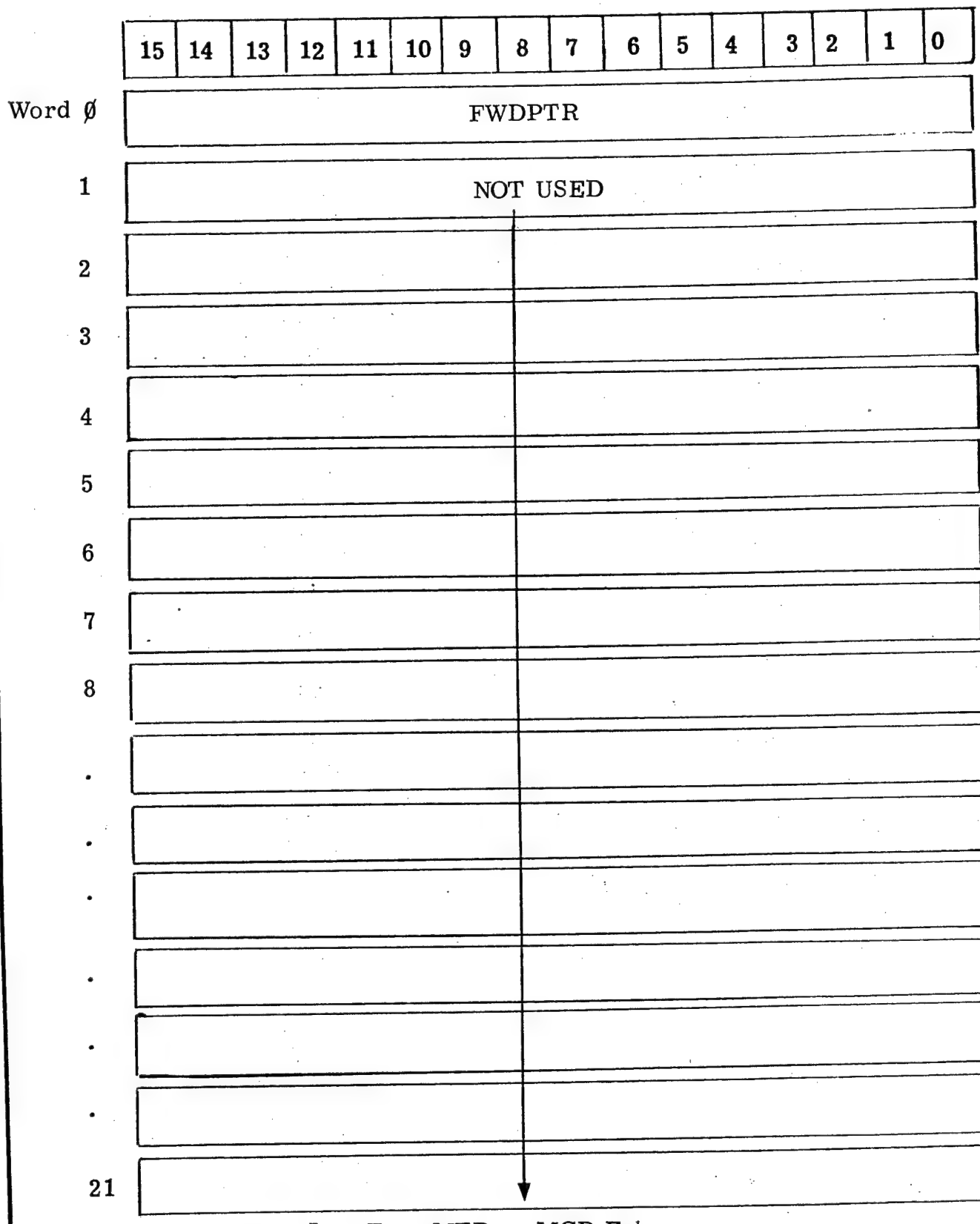



Figure 5a. Free MTB or MCB Entry

| Field | Description | Units | LSB |
|---|---|---|---|
| FWDPTR | Address of Next Entry in Queue or Ø, if this is EOQ Entry | N/A | N/A |

Figure 5b. Free MTB or MCB entry.

3.3.7.3 Free MCB Queue

a) Purpose: - The Free MCB Queue shall be used to hold the surplus of memory blocks used for dynamic interprocessor storage allocation. MCB's shall be used for interprocessor message storage. Initially, this queue shall contain all memory blocks available for interprocessor use.

b) Queue Definition Variables: -

1) RMP/CP Free MCB Queue - RCSTR (= CRSTR) shall be the SOQ pointer. RCEND (= CREND) shall be the EOQ pointer. RCCOM (=CRCOM) shall be the low memory address of the memory allocated for RMP/CP MCB's.

2) RMP/AP Free MCB Queue - RASTR (=ARSTR) shall be the SOQ pointer. RAEND (=AREND) shall be the EOQ pointer. RACOM (= ARCOM) shall be the low memory address of the memory allocated for RMP/AP MCB's.

3) CP/AP Free MCB Queue - CASTR (= ACSTR) shall be the SOQ pointer. CAEND (= ACEND) shall be the EOQ pointer. CACOM (= ACCOM) shall be the low memory address of the memory allocated for CP/AP MCB's.

c) Entry Format: -

See Figure 5.

3.3.7.4 Interprocessor Message Transfer Queue

a) Purpose: - This queue shall provide the capability to bundle messages during the interprocessor communication process. This communication shall be conducted on a polling (non-interrupt) basis and these queues provide multibuffering of messages. There shall be six of these queues:

1) RMP to CP
2) CP to RMP
3) RMP to AP
4) AP to RMP
5) CP to AP
6) AP to CP

Initially, these queues shall be empty.

b) Queue Definition Variables: - For each of the six queues there shall be a polling flag (RCPOL, CRPOL, RAPOL, ARPOL, CAPOL, and ACPOL, respectively) and a chain pointer (RCPTR, CRPTR, RAPTR, ARPTR, CAPTR, and ACPTR, respectively). The polling flag shall be a count of the messages in the queue or Ø if it is empty. The poll is incremented by the sender and decremented by the receiver. The chain pointer shall be the address of the first MCB in the queue.

c) Entry Format: -
See Figure 4.

## 3.4 INPUT/OUTPUT FORMATS

Executive message formats used for intraprocessor and interprocessor communication shall be as specified in the CDBDD, 53959-GT-0751. Sorter message formats shall be as specified in the System Controller - Sorter ICD, 53959-JK-1002.

## 3.5 REQUIRED SYSTEM LIBRARY SUBROUTINES

None.

## 3.6 CONDITIONS FOR INITIALIZATION

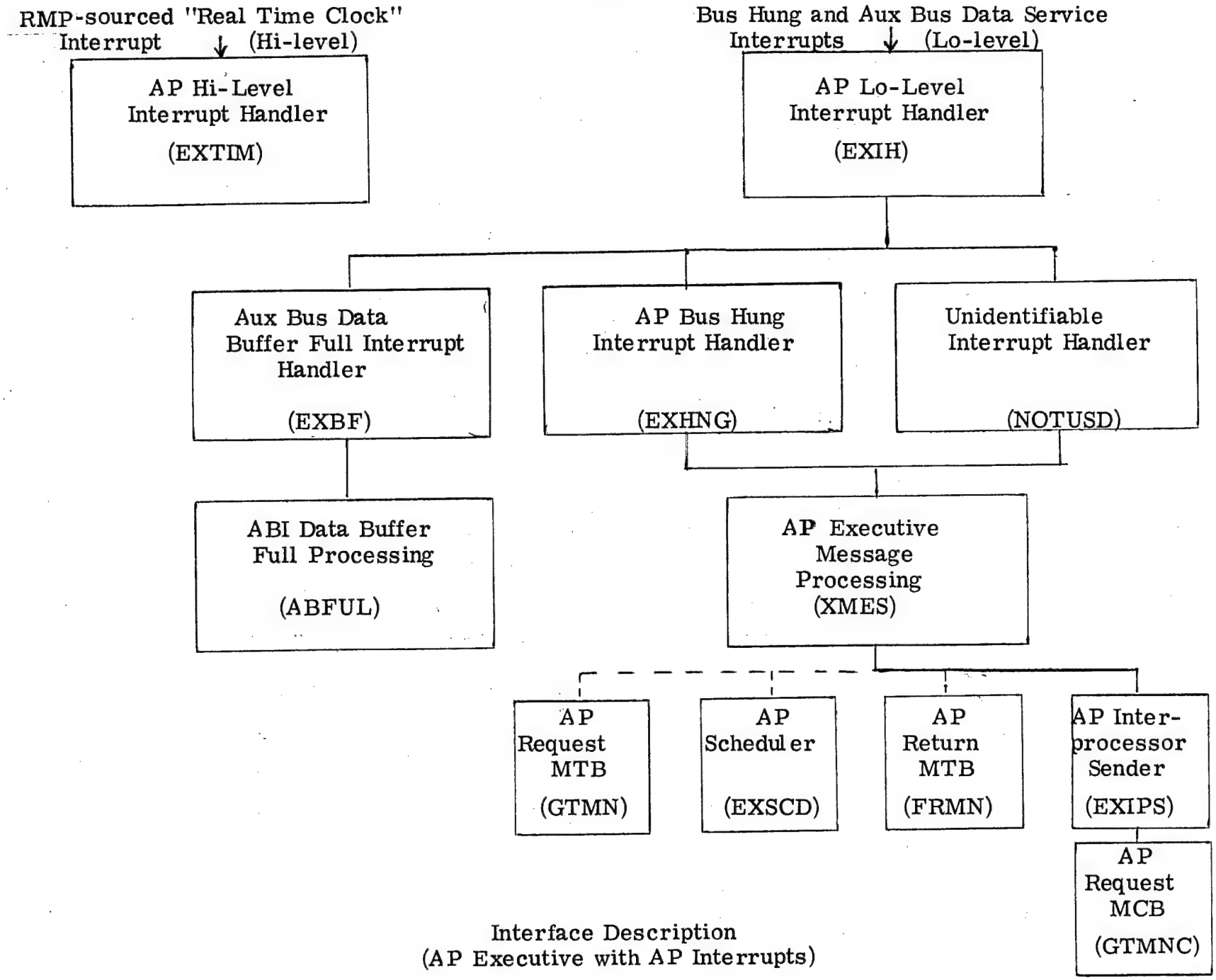
The RMP, CP, and AP shall perform all initialization of IEWS, with the exception of the initialization of the circular buffer used as a depository for STE-destined messages.

## 3.7 SUBPROGRAM LIMITATIONS

An excessive backlog of ECM processing requests will result in the exhaustion of the pool of MTB storage blocks and in effect will terminate all processing in the processor in which overloading occurred. No attempt at recovery shall be made in IEWS, ADM.

## 3.8 INTERFACE DESCRIPTION

The following diagrams describe the interface of the Executive with other IEWS operational software and also the relationship of the various Executive subfunctions.




Interface Description
(AP Executive with AP Interrupts)

RAYTHEON

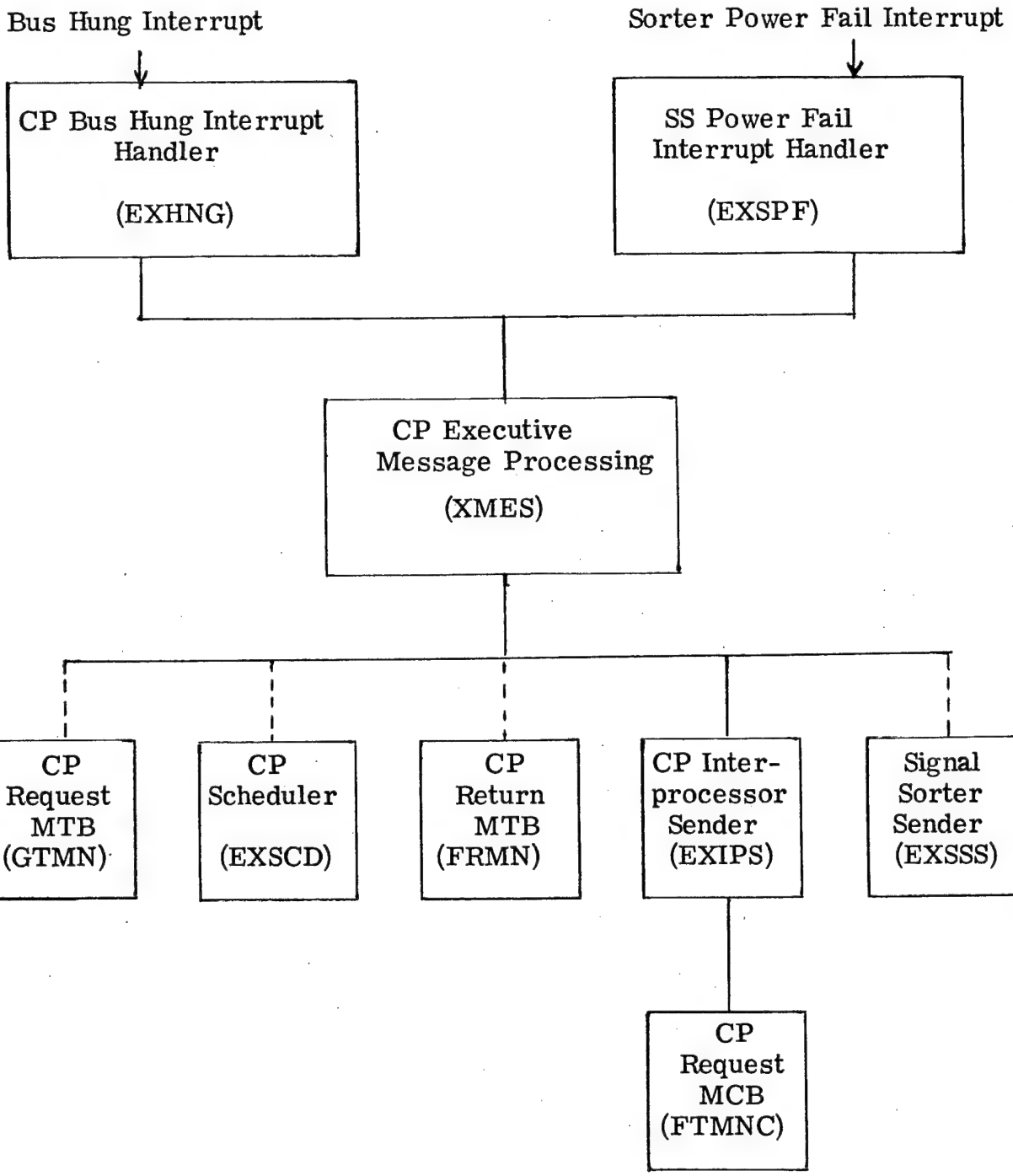
RAYTHEON COMPANY
LEXINGTON, MASS. 02173

CODE IDENT NO. 49956

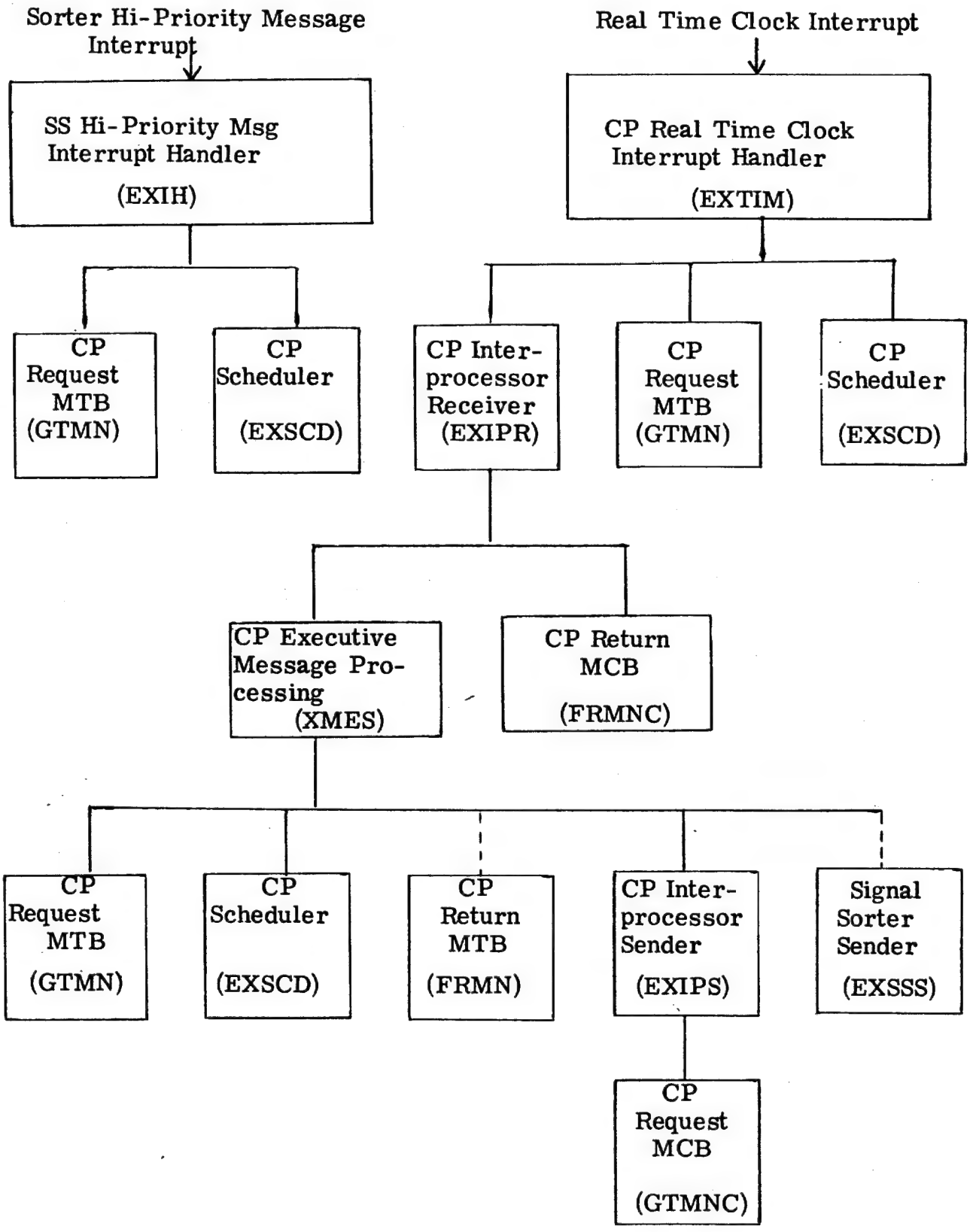
SPEC NO. 53959-GT-0756

SHEET 70 OF 78

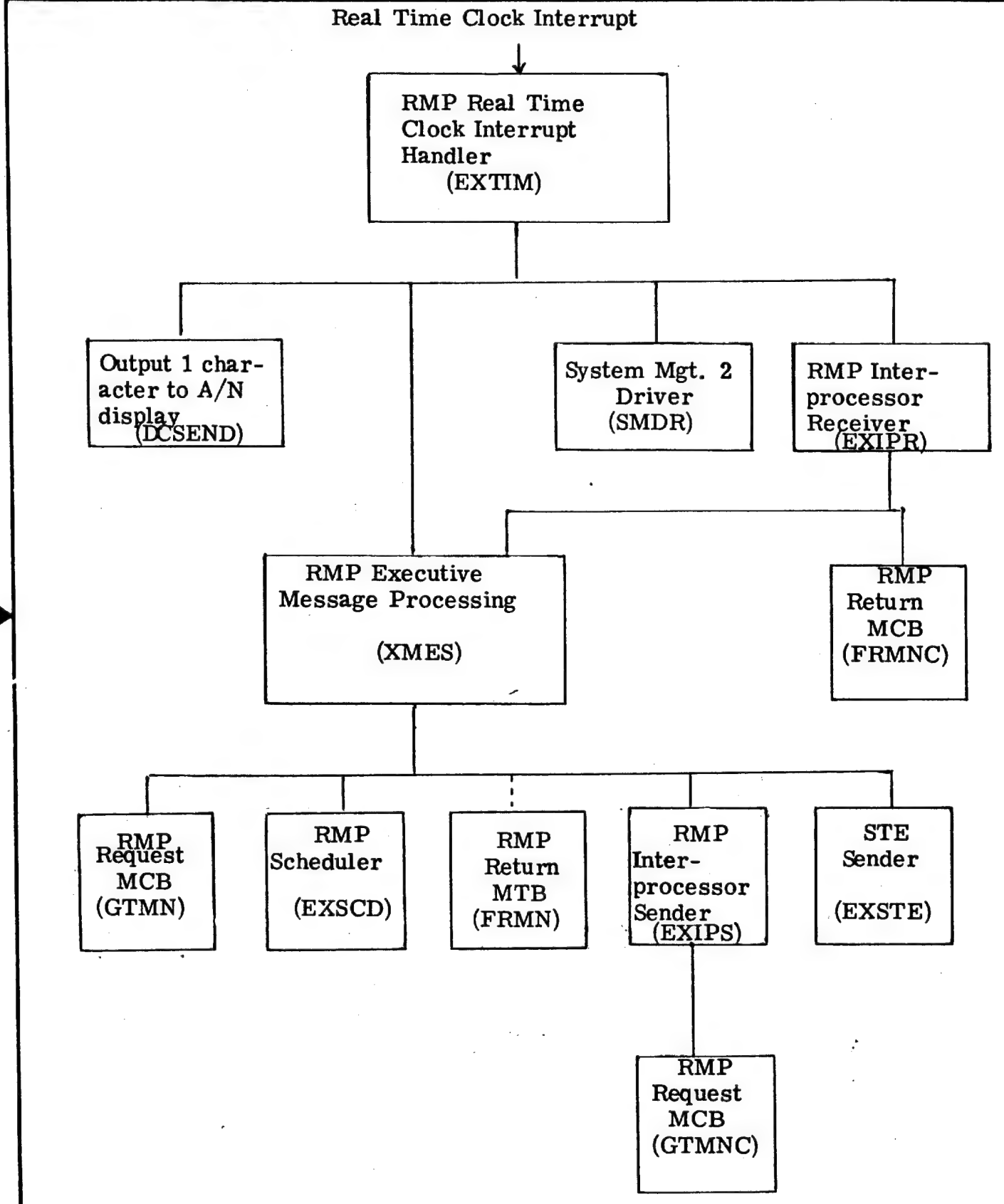
REV A



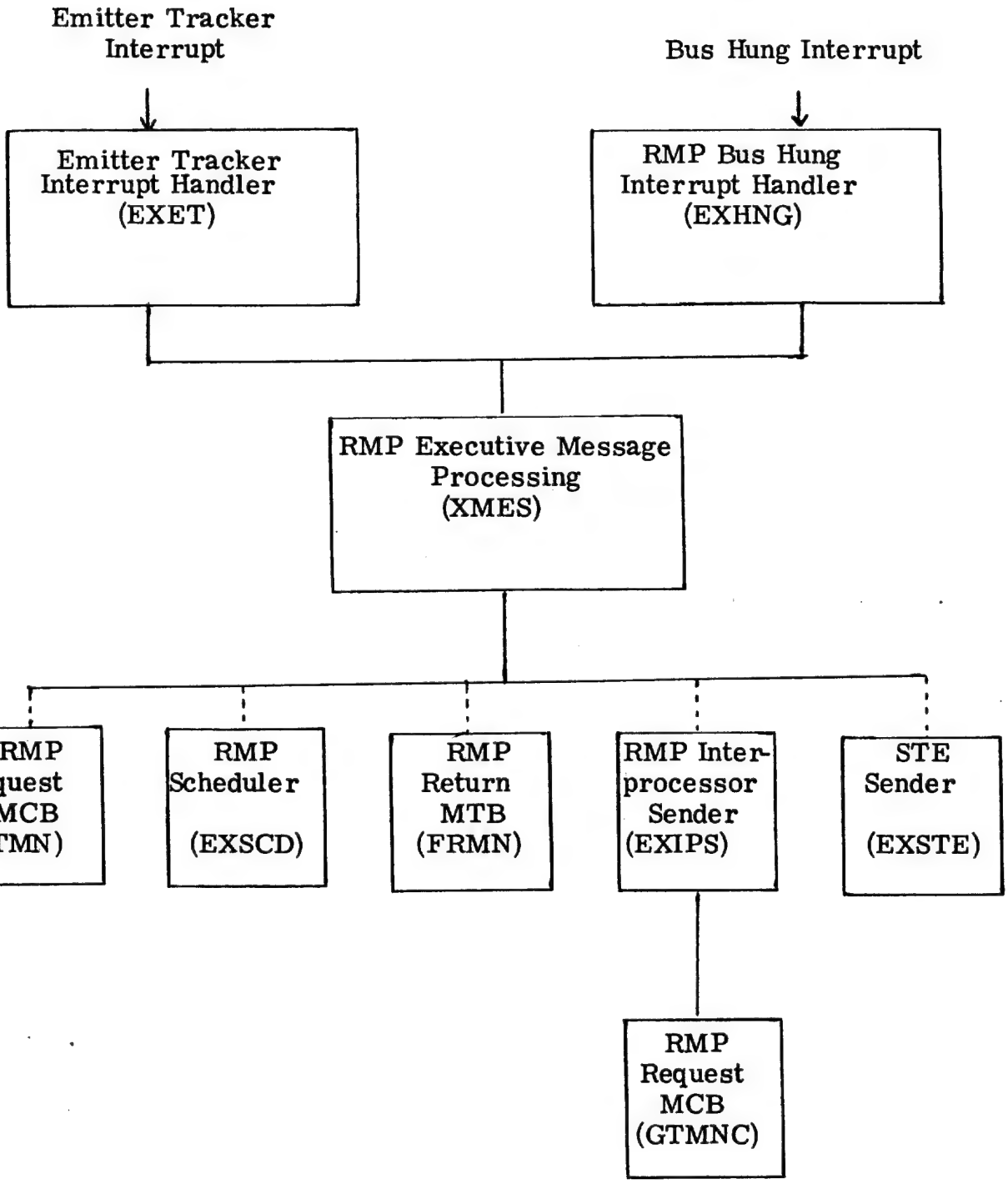

Interface Description
(CP Executive with CP Interrupts)



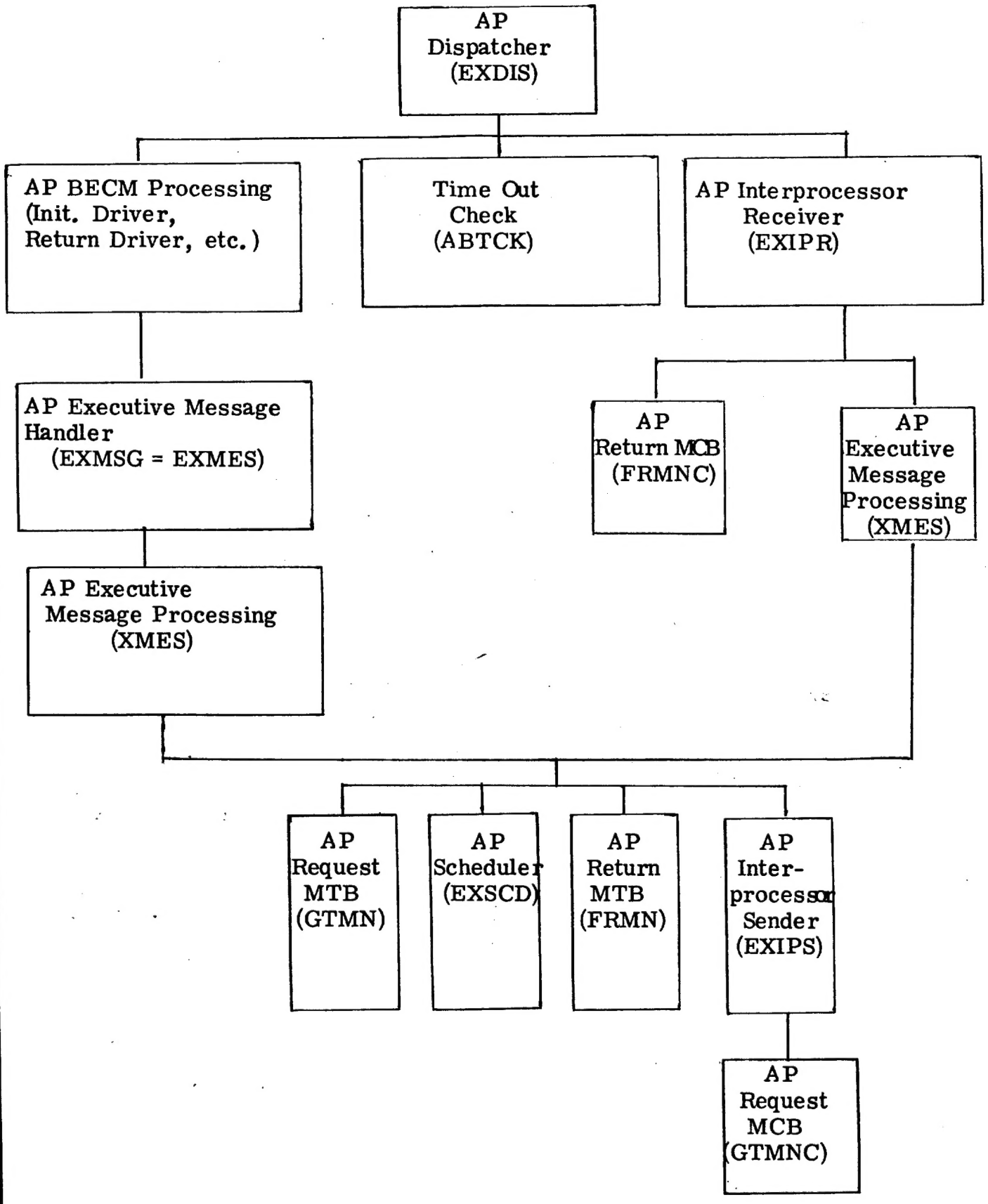

Interface Description
(CP Executive with CP Interrupts)



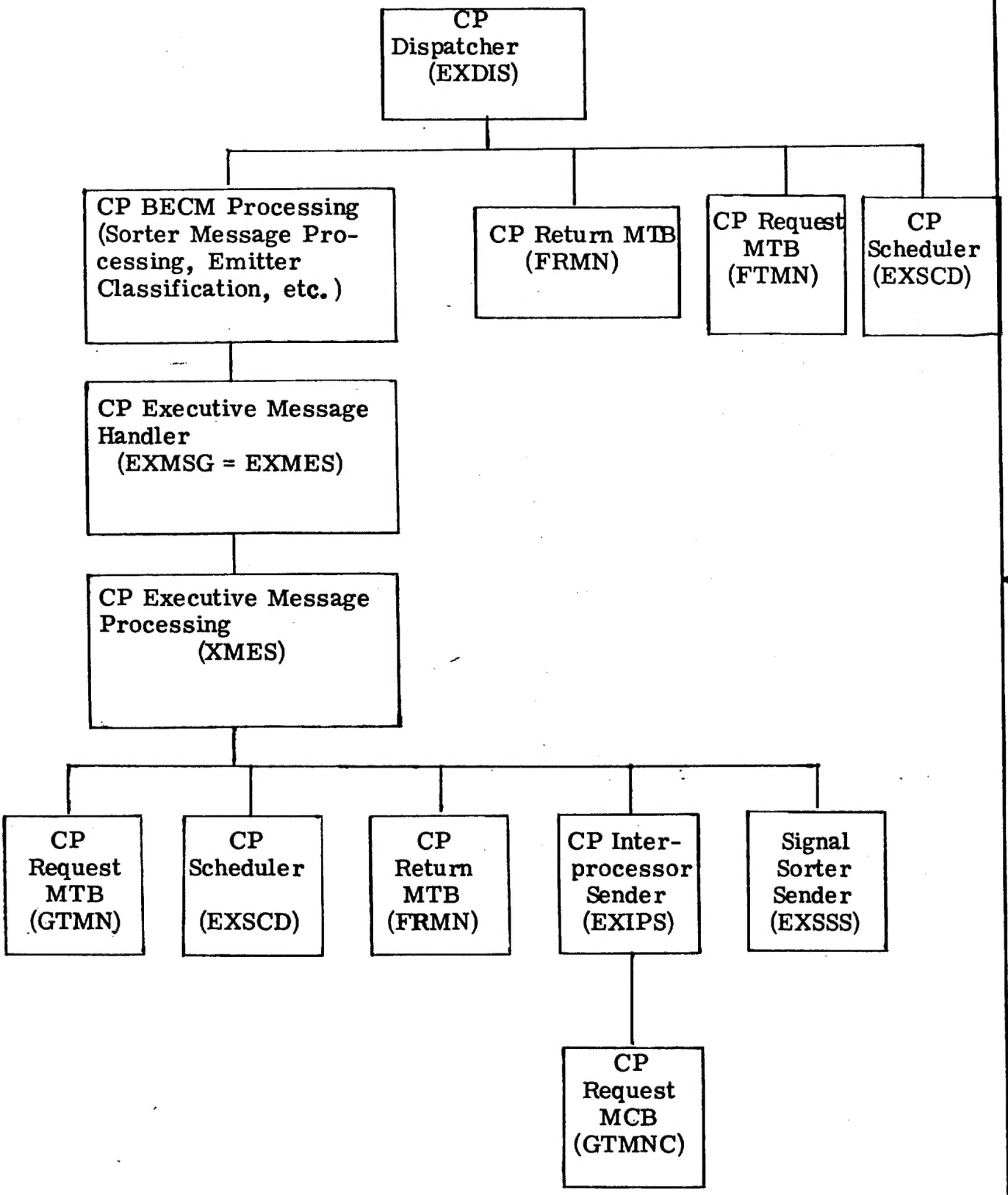

Interface Description
(RMP Executive with RMP Interrupts)



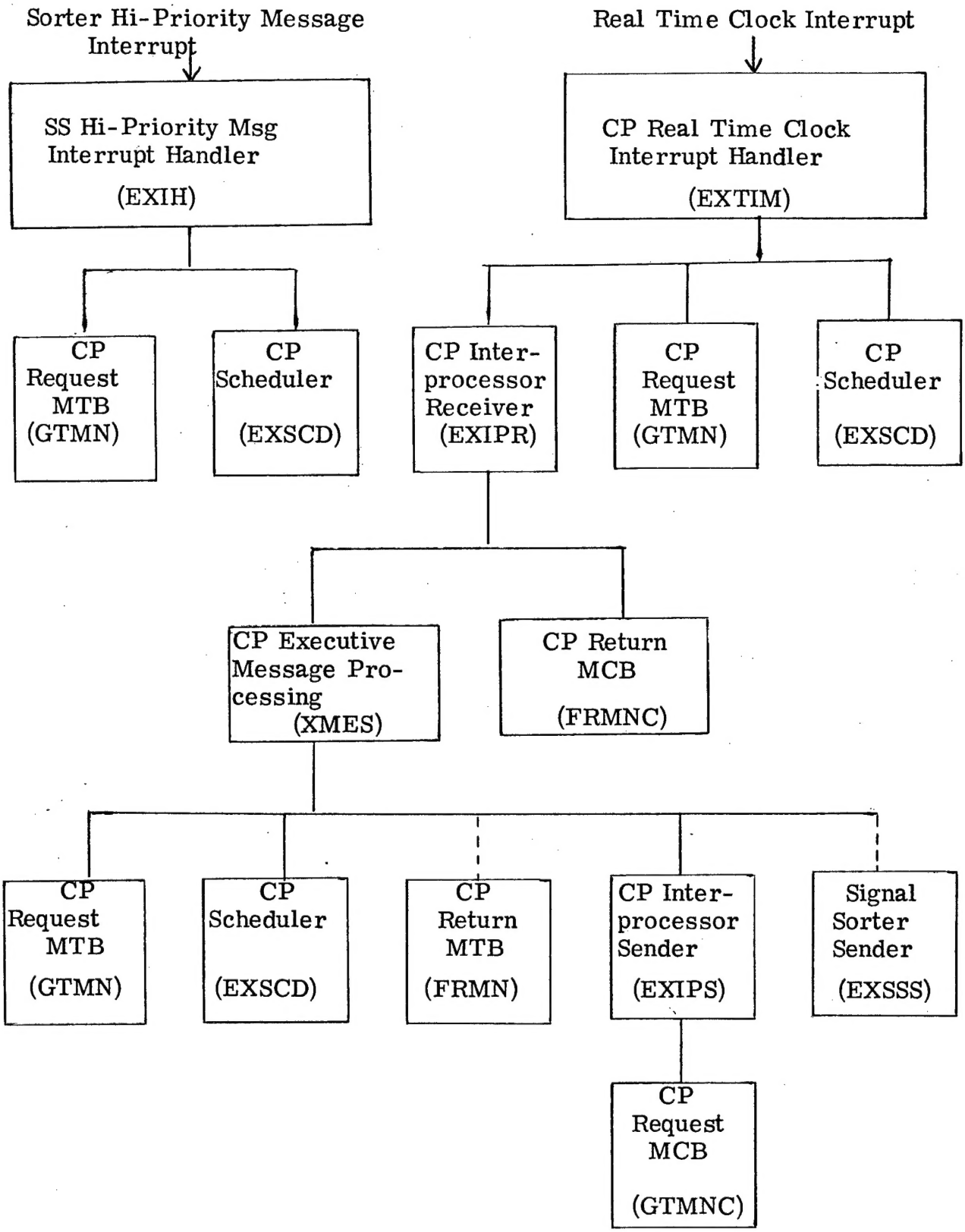

Interface Description
(RMP Executive with RMP Interrupts)



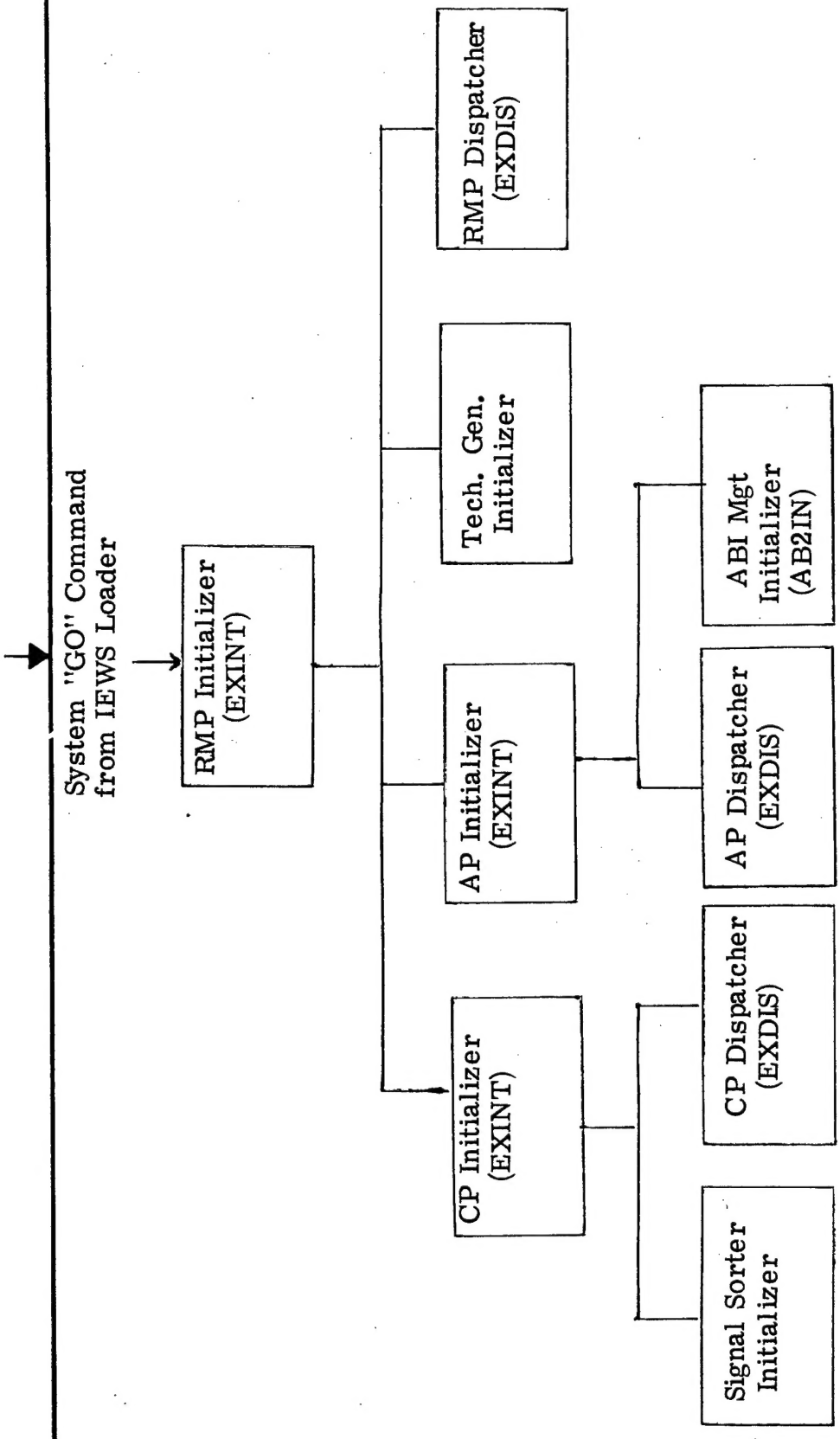

Interface Description
(AP Executive with AP BECM, RMP EXEC, and CP EXEC)




Interface Description
(CP Executive with CP BECM, RMP EXEC, AP EXEC and SS)




Interface Description
(CP Executive with CP Interrupts)




Interface Description (System Initialization)